197-053

1

# DOCUMENTATION OF DECISION-AIDING SOFTWARE: EVAL SYSTEM SPECIFICATION

DECISIONS AND DESIGNS INC.

Linda B. Allardyce
Dorothy M. Amey
Phillip H. Feuerwerger
Roy M. Gulick

November 1979

N00014-79-C-0069

ADVANCED ARPA
DECISION TECHNOLOGY
PROGRAM

CYBERNETICS TECHNOLOGY OFFICE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
Office of Naval Research • Engineering Psychology Programs

82 11 26 193

# DOCUMENTATION OF DECISION-AIDING SOFTWARE:
## EVAL SYSTEM SPECIFICATION

by

Linda B. Allardyce, Dorothy M. Amey, Phillip H. Feuerwerger, and Roy M. Gulick

Sponsored by

Defense Advanced Research Projects Agency
ARPA Order 3469

November 1979

DECISIONS and DESIGNS, INC.
Suite 600, 8400 Westpark Drive
P.O.Box 907
McLean, Virginia 22101
(703) 821-2828

## CONTENTS

| | Page |
|---|---|
| FIGURES | iii |
| 1.0 INTRODUCTION | 1 |
| 1.1 Purpose of the System Specification | 1 |
| 1.2 References | 1 |
| 1.3 Terms | 2 |
| 1.3.1 EVAL | 2 |
| 1.3.2 HIPO | 2 |
| 2.0 DESIGN DETAILS | 3 |
| 2.1 Background | 3 |
| 2.2 General Operating Procedures | 3 |
| 2.3 System Logical Flow | 3 |
| 2.4 HIPO Documentation | 6 |

## FIGURES

| Figure | | Page |
|---|---|---|
| 2-1 | LEGEND OF HIPO SYMBOLS | 5 |
| 2-2 | EVAL SYSTEM OVERVIEW | 7 |
| 2-3 | STRUCTURE VISUAL TABLE OF CONTENTS | 8 |
| 2-4 | RUN VISUAL TABLE OF CONTENTS | 9 |

# EVAL SYSTEM SPECIFICATION

## 1.0 INTRODUCTION

### 1.1 Purpose of the System Specification

The EVAL System Specification is a technical document written for software development personnel. Together with the EVAL Functional Description, it guides the software development effort by identifying the functional requirements and by providing structured logic diagrams that depict the flow, control, and processing of information within the system.

The System Specification is generic and is intended to guide and facilitate the preparation of the language-specific and computer hardware-specific documentation and coding that are necessary to implement and operate EVAL at an installation.

### 1.2 References

1.2.1 IBM, HIPO--A Design Aid and Documentation Technique. Technical Publication GC20-1851-0. White Plains, New York: IBM, October 1974.-

1.2.2 Allardyce, Linda B.; Amey, Dorothy M.; Feuerwerger, Phillip H.; Gulick, Roy M. Documentation of Decision-Aiding Software: EVAL Functional Description. McLean, Virginia: Decisions and Designs, Incorporated, November 1979.

1.2.3 Allardyce, Linda B.; Amey, Dorothy M.; Feuerwerger, Phillip H.; Gulick, Roy M. Documentation of Decision-Aiding Software: EVAL Users Manual. McLean, Virginia: Decisions and Designs, Incorporated, November 1979.

## 1.3 Terms

1.3.1 EVAL - EVAL is an abbreviation for evaluation, reflecting the system's major area of applicability.

1.3.2 HIPO - The specification uses the standard Hierarchy plus Input-Process-Output (HIPO) diagramming technique to depict the structural design and logical flow of the system. A legend explaining the HIPO diagramming symbols is included. Reference 1.2.1 provides a complete description of the HIPO documentation technique.

# 2.0 DESIGN DETAILS

## 2.1 Background

Systems development personnel should refer to the EVAL Functional Description, Reference 1.2.2, in conjunction with the documentation contained in this specification. The Functional Description details the evaluation models implemented by EVAL and discusses the specific functions that the software must perform. In addition, systems development personnel may wish to refer to the EVAL User's Manual, Reference 1.2.3.

## 2.2 General Operating Procedures

EVAL is a menu-driven system. That is, the system is designed to interact with the user by presenting a sequential hierarchy of menus and asking the user to respond by selecting one option from the current menu. If the user does not select one of the menu options, the system displays the previous menu. In this manner, the user moves up and down the hierarchy, as desired. Whenever data entry is required as a result of option selection, the system specifically requests the data and specifies the format.

The system is also designed to be generally forgiving of procedural errors by the user.

## 2.3 System Logical Flow

EVAL is a hierarchically structured, modular software system. The system structure and logical flow lends itself to presentation in the form of HIPO diagrams, which are contained in this document.

The main purpose of the HIPO diagrams is to provide, in a pictorial manner, the complete set of modular elements necessary to the operation of EVAL, including all input, output, and internal functional processing. This is done by displaying the input items necessary to the process step which uses them, defining the process, and showing the resulting output of the process step.

The HIPO documentation diagrams are designed and drawn in a hierarchical fashion from the main calling routines to the detail-level operation/calculation routines. Extended written descriptions are given below a HIPO diagram whenever it is deemed necessary.

A complete description and explanation of the symbolic notation used in the HIPO diagrams is given in Reference 1.2.1. An abbreviated legend for the symbols used in this specification is given in Figure 2-1. Note that:

a. External subroutines are depicted partly in the Process block and partly out. Internal subroutines are always shown within the Process block.

b. Overview diagrams show general inputs and outputs only, whereas detail/subroutine-level diagrams show specific input/output tables and/or displays.

c. Rectangular boxes inside the Input/Output block areas are generally used to denote single data items. Two or more boxes are grouped to show that several data items are input/output.

d. Rectangular boxes inside the Process block indicate repetitive subprocesses.

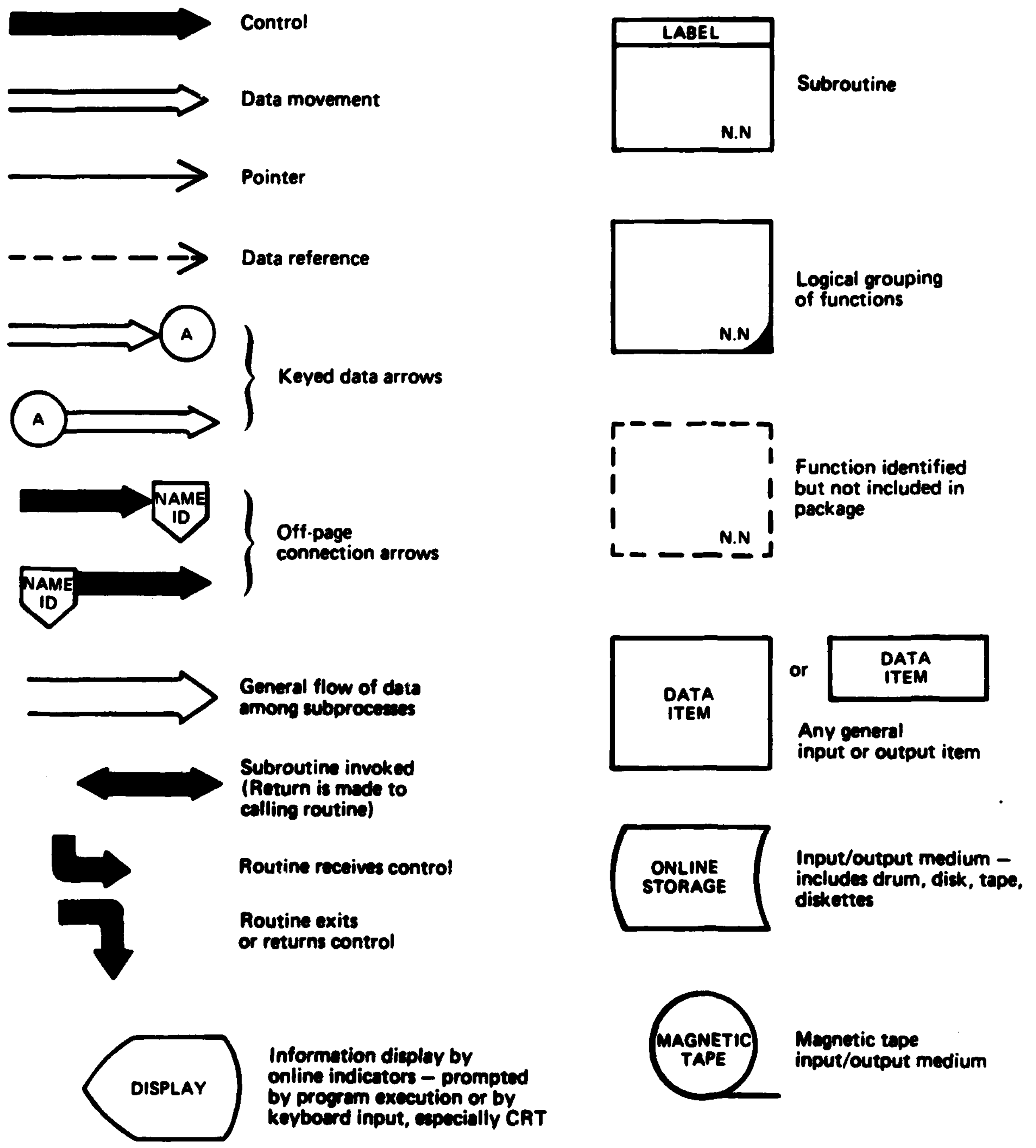

Control
Data movement
Pointer
Data reference
A
Keyed data arrows
NAME ID
Off-page connection arrows
General flow of data among subprocesses
Subroutine invoked (Return is made to calling routine)
Routine receives control
Routine exits or returns control
DISPLAY
Information display by online indicators – prompted by program execution or by keyboard input, especially CRT
LABEL
N.N
Subroutine
Logical grouping of functions
Function identified but not included in package
DATA ITEM
or
Any general input or output item
ONLINE STORAGE
Input/output medium – includes drum, disk, tape, diskettes
MAGNETIC TAPE
Magnetic tape input/output medium

**Figure 2-1**
**LEGEND OF HIPO SYMBOLS**

The HIPO diagrams appear in the next section, which completes the system specification.

## 2.4 HIPO Documentation

The HIPO diagram identification numbers and figure numbers used in this section stand alone; i.e., they start with 1.0, increase hierarchically, and are independent of the numbering scheme used to this point in this document.

The EVAL software consists of two separate subsystems: STRUCTURE and RUN. Figure 2-2 is the system overview chart. The STRUCTURE subsystem is used to create a new evaluation structure or to revise an existing structure. The RUN subsystem is used to specify importance weights and utilities and to display the results of an evaluation model.

Figure 2-3 is a subsystem structure chart for the STRUCTURE subsystem. It represents the overall program logic flow in a visual table of contents. The Visual Table of Contents diagram shows the hierarchical structure, the functional description labels, and the diagram (chart) identifiers of the functions performed by STRUCTURE. Similarly, Figure 2-4 is a visual table of contents diagram for the RUN subsystem.

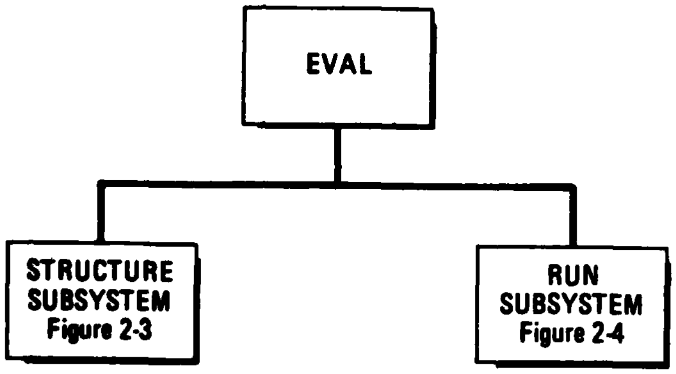


Figure 2-2
EVAL SYSTEM OVERVIEW

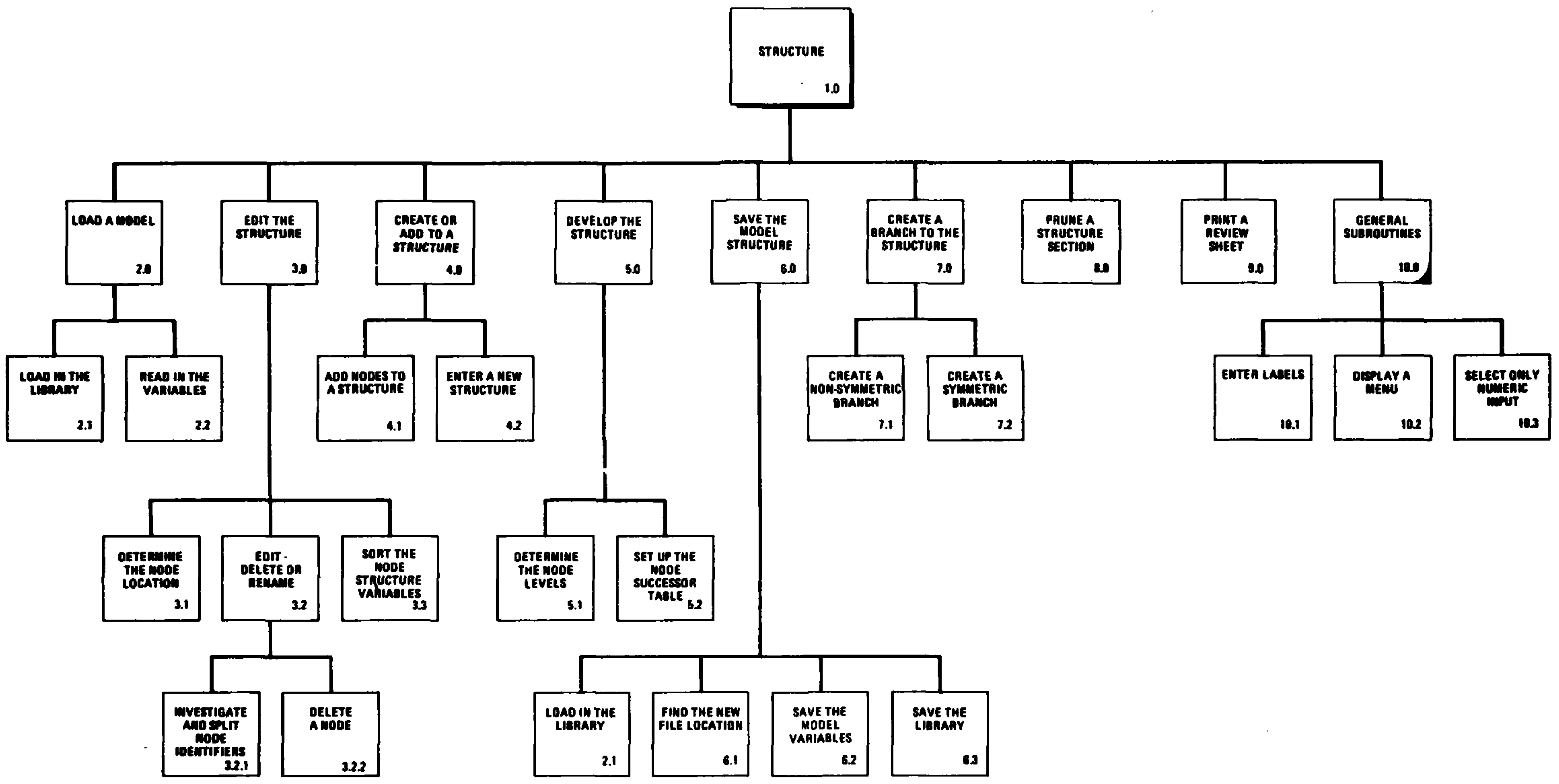

STRUCTURE 1.0
LOAD A MODEL 2.0
LOAD IN THE LIBRARY 2.1
READ IN THE VARIABLES 2.2
EDIT THE STRUCTURE 3.0
DETERMINE THE NODE LOCATION 3.1
EDIT - DELETE OR RENAME 3.2
INVESTIGATE AND SPLIT NODE IDENTIFIERS 3.2.1
DELETE A NODE 3.2.2
SORT THE NODE STRUCTURE VARIABLES 3.3
CREATE OR ADD TO A STRUCTURE 4.0
ADD NODES TO A STRUCTURE 4.1
ENTER A NEW STRUCTURE 4.2
DEVELOP THE STRUCTURE 5.0
DETERMINE THE NODE LEVELS 5.1
SET UP THE NODE SUCCESSOR TABLE 5.2
SAVE THE MODEL STRUCTURE 6.0
LOAD IN THE LIBRARY 2.1
FIND THE NEW FILE LOCATION 6.1
SAVE THE MODEL VARIABLES 6.2
SAVE THE LIBRARY 6.3
CREATE A BRANCH TO THE STRUCTURE 7.0
CREATE A NON-SYMMETRIC BRANCH 7.1
CREATE A SYMMETRIC BRANCH 7.2
PRUNE A STRUCTURE SECTION 8.0
PRINT A REVIEW SHEET 9.0
GENERAL SUBROUTINES 10.0
ENTER LABELS 10.1
DISPLAY A MENU 10.2
SELECT ONLY NUMERIC INPUT 10.3

Figure 2-3

STRUCTURE VISUAL TABLE OF CONTENTS

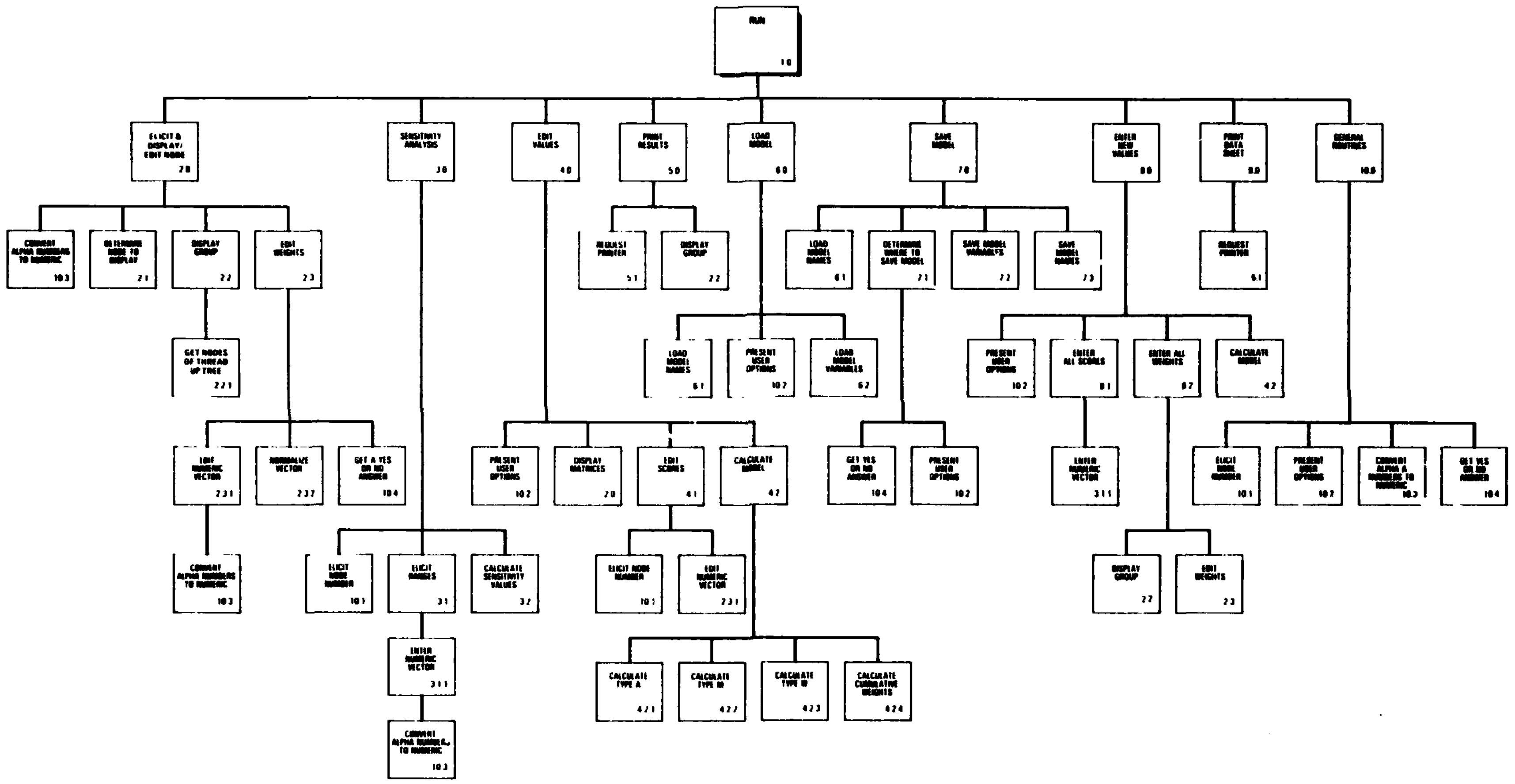
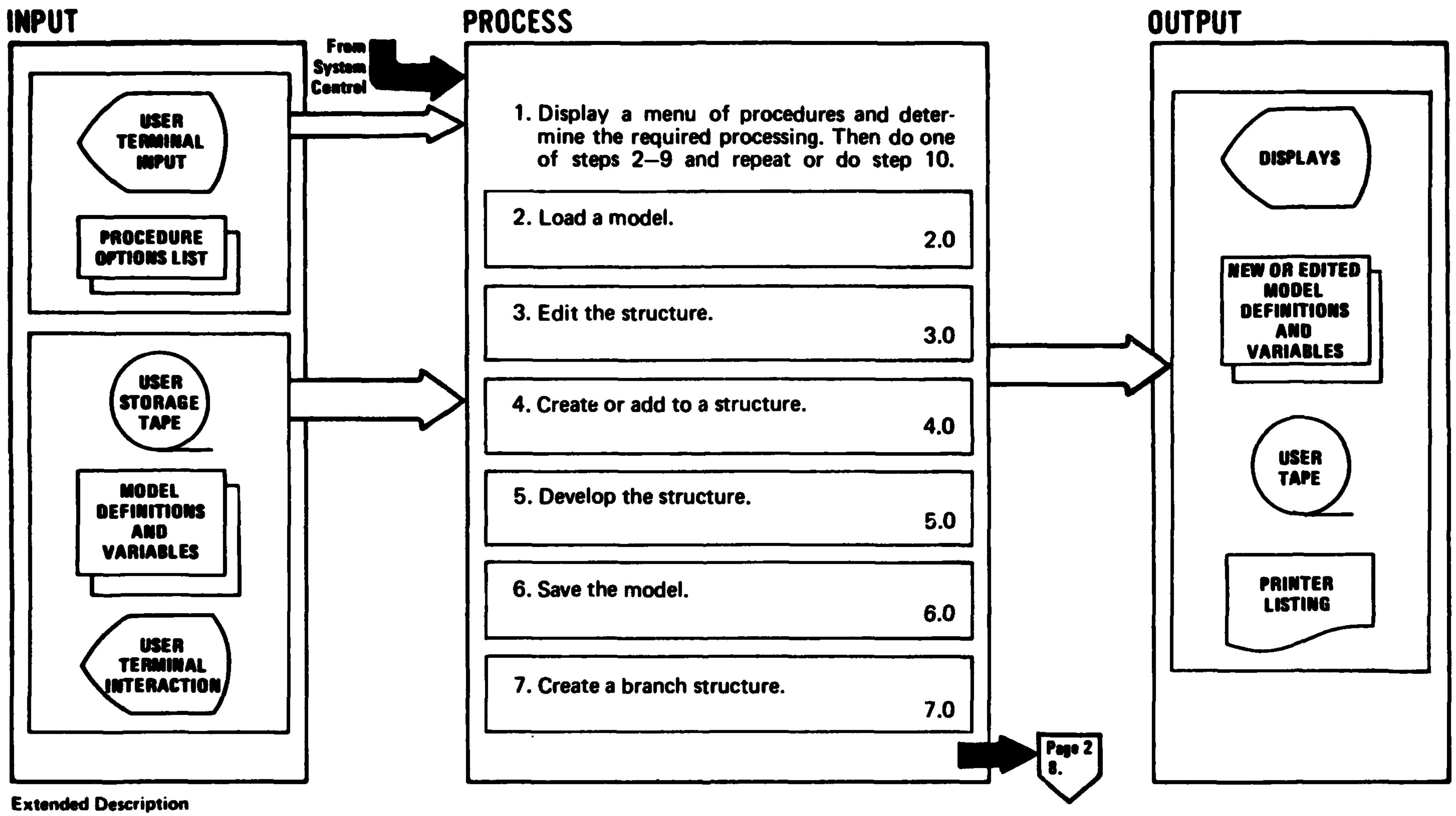


Figure 2-4
RUN VISUAL TABLE OF CONTENTS

System/Program: STRUCTURE Name:

Diagram ID: 1.0 Description: Main Routine Page: 1 of 2



INPUT
USER TERMINAL INPUT
PROCEDURE OPTIONS LIST
USER STORAGE TAPE
MODEL DEFINITIONS AND VARIABLES
USER TERMINAL INTERACTION
From System Control
PROCESS
1. Display a menu of procedures and determine the required processing. Then do one of steps 2–9 and repeat or do step 10.
2. Load a model. 2.0
3. Edit the structure. 3.0
4. Create or add to a structure. 4.0
5. Develop the structure. 5.0
6. Save the model. 6.0
7. Create a branch structure. 7.0
Page 2 8.
OUTPUT
DISPLAYS
NEW OR EDITED MODEL DEFINITIONS AND VARIABLES
USER TAPE
PRINTER LISTING

**Extended Description**

The list of program procedures is displayed so that the user may select the next process to be performed. The list is displayed in menu format which allows the association of position numbers with different options in the list.

1. The user is prompted for a choice of operations. The chosen procedure is invoked via one of steps 2–9. If the user responds with blank or null input, then step 10 is executed.

2. The existence of EVAL/STRUCTURE models on tape (storage) is determined and a selected model is read.

3. The structure (or model) currently defined by the program variables may be

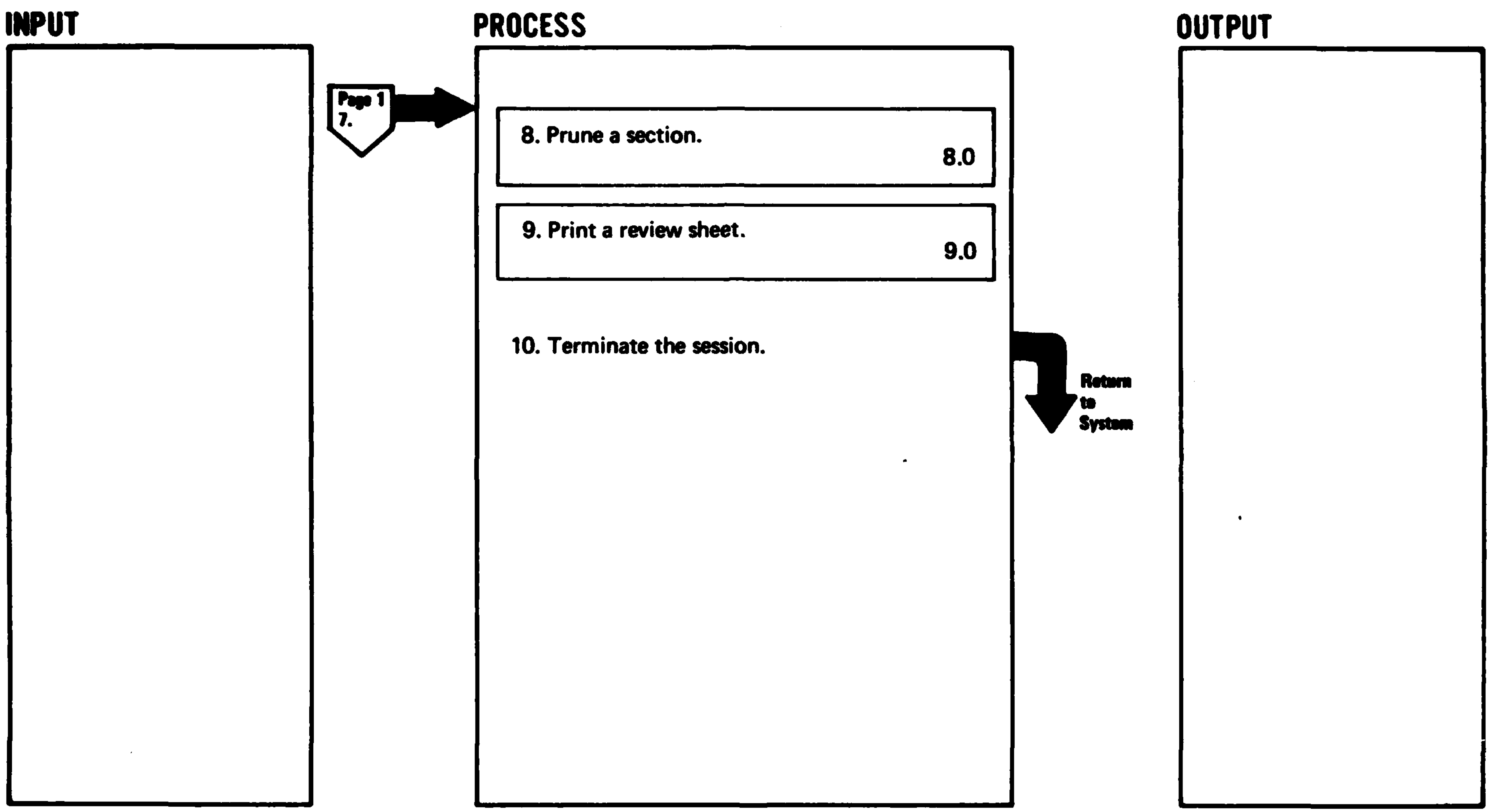
4. A new structure may be entered via user interaction or nodes may be added to an existing structure.

5. This step causes the completion of the model structure by setting up variables which interface with the RUN program. This step should always be performed before step 6.

6. The currently defined model structure may be stored via this step.

7. A branch or subtree may be defined and later added to a structure in procedure 4.

| | |
|---|---|
| System/Program: STRUCTURE | Name: |
| Diagram ID: 1.0 Description Main Routine | Page: 2 of 2 |



**Extended Description**

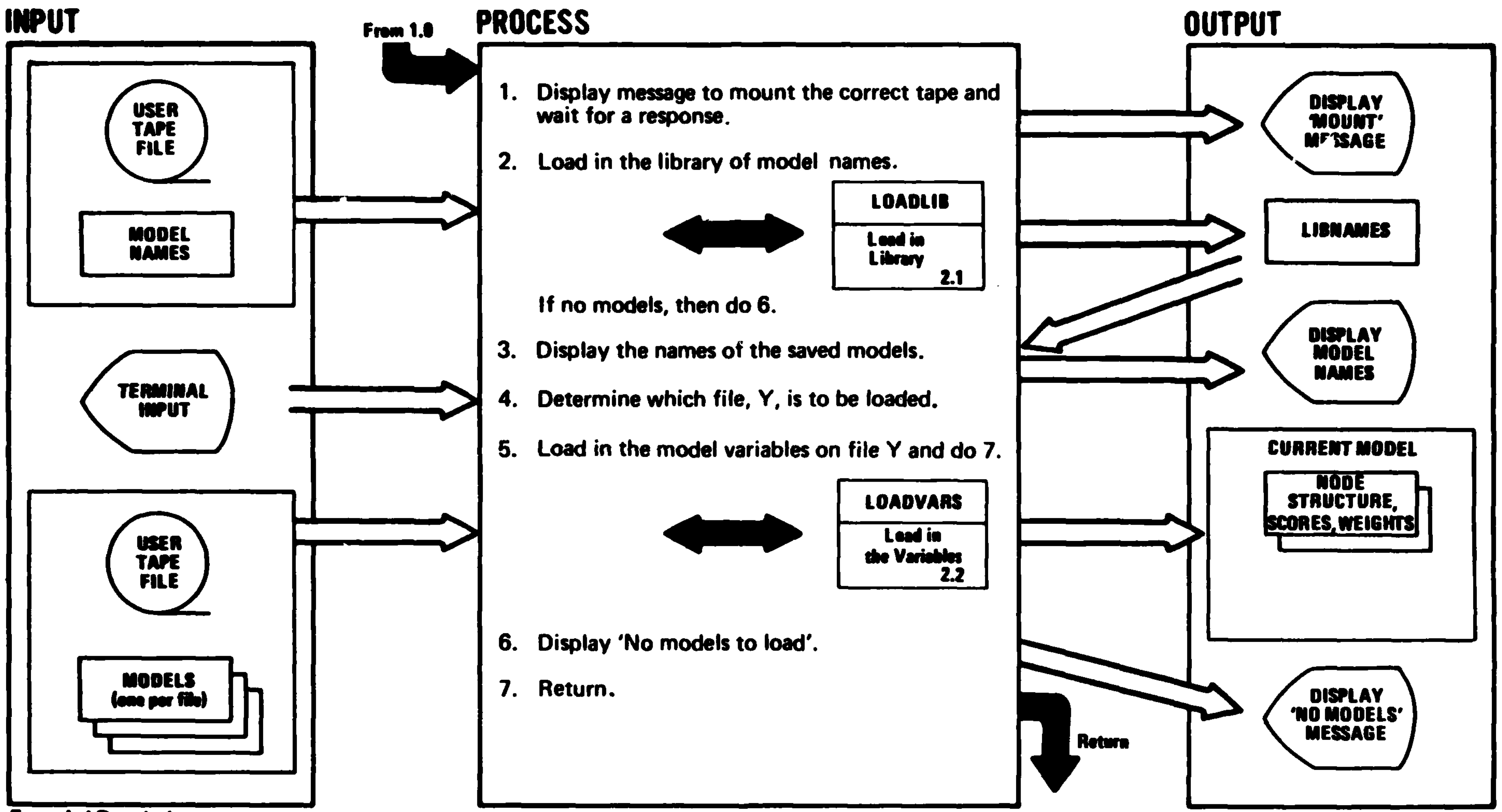
8. Groups of nodes may be deleted from the currently defined structure.

9. A printout of the structure as it is currently defined is obtained.

10. The program ends here: a restart option will cause step 1 to be executed again. When a session is terminated, all branch structures or subtrees defined are deleted.

System/Program: STRUCTURE Name:

Diagram ID: 2.0 Description Load A Model Page: of



1. Display message to mount the correct tape and wait for a response.
2. Load in the library of model names.

   If no models, then do 6.
3. Display the names of the saved models.
4. Determine which file, Y, is to be loaded.
5. Load in the model variables on file Y and do 7.
6. Display 'No models to load'.
7. Return.

**Extended Description**

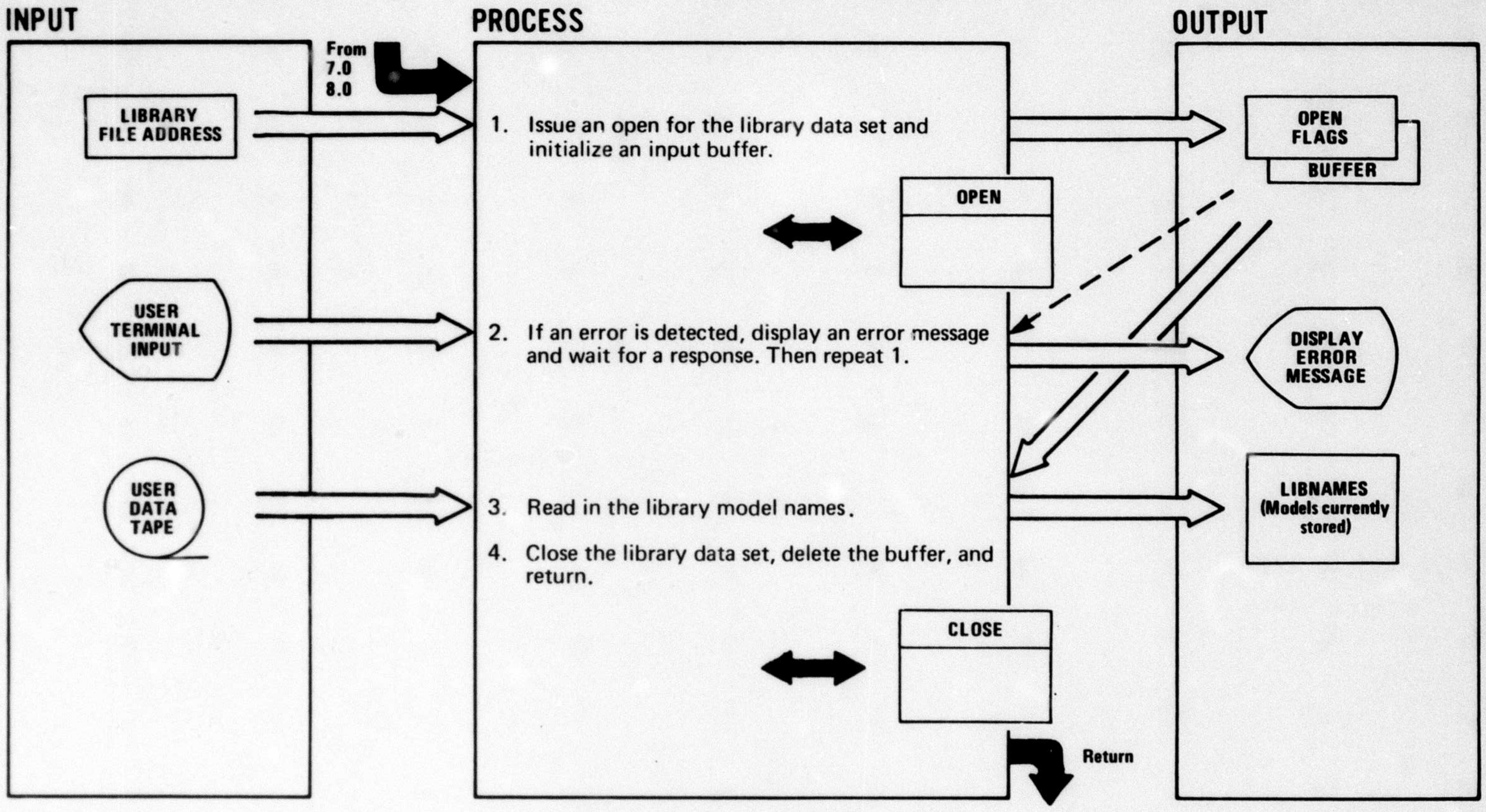
1. The user may have many tape files on which formatted models are stored. In this step, the user is prompted for a response indicating the desired tape is mounted and online.

3. The names of the models existing on the mounted tape are displayed in list or MENU format so that the user may select a model for loading.

4. The user is prompted for a model selection: the response may be the list item number or the model name. The requested model is stored in the same tape file as its position relative to the other model names in the displayed list.

System/Program: STRUCTURE Name: LOADLIB

Diagram ID: 2.1 Description Load in the Library Page: ___ of ___



INPUT
PROCESS
OUTPUT
From 7.0 8.0
LIBRARY FILE ADDRESS
USER TERMINAL INPUT
USER DATA TAPE
1. Issue an open for the library data set and initialize an input buffer.
OPEN
2. If an error is detected, display an error message and wait for a response. Then repeat 1.
3. Read in the library model names.
4. Close the library data set, delete the buffer, and return.
CLOSE
Return
OPEN FLAGS
BUFFER
DISPLAY ERROR MESSAGE
LIBNAMES (Models currently stored)

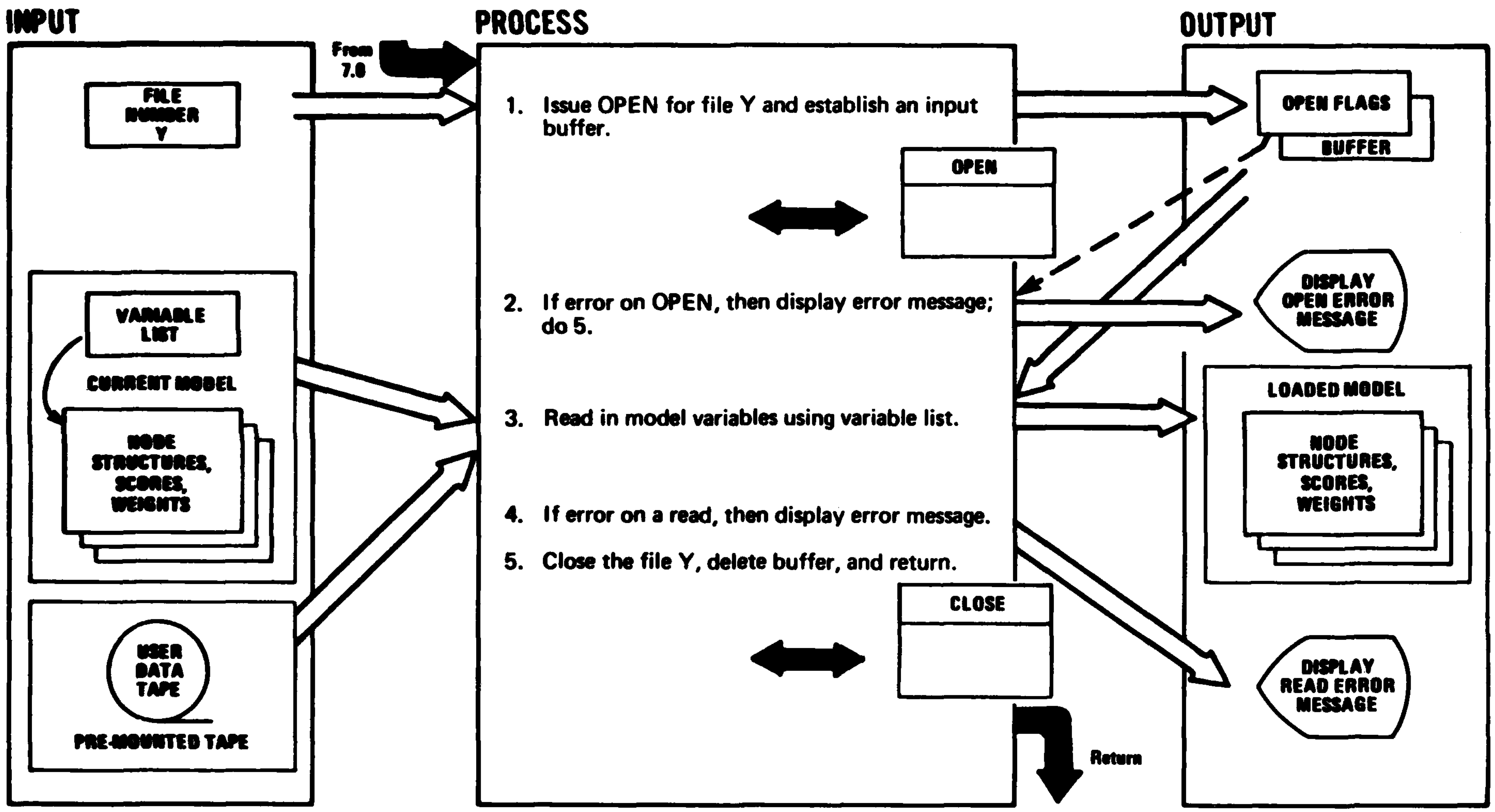
**Extended Description**

2. The library file of model names is available on each formatted data tape. The file is usually stored and retrieved as a character array and resides on the same device with model data and structure variables. A system OPEN command is needed to ensure that the data file is online and accessible for reading. An input buffer is needed and provides the link between stored information and program addressable information.

3. The library model names are retrieved from storage. The character array used for holding these model names, LIBNAMES, is of a form which facilitates display; thus, the names may all be of equal character length.

4. A system CLOSE command is issued to free the data file for later use.

| | |
|---|---|
| System/Program: STRUCTURE | Name: LOADVARS |
| Diagram ID: 2.2 | Description Read in the Variables |
| Page: 1 of 3 | |



1. Issue OPEN for file Y and establish an input buffer.
2. If error on OPEN, then display error message; do 5.
3. Read in model variables using variable list.
4. If error on a read, then display error message.
5. Close the file Y, delete buffer, and return.

**Extended Description**

3. A list of variable Names or identifiers is kept so that load and store routines will always process the variables in the same sequence order.

4. The Model variables retrieved from storage are used in all other program functions (see Diagram 1.0). The variables which must be loaded are the following:

- ●OUTLINE TABLE
- ●NODE LABLES
- ●SCORES
- ●WEIGHTS
- ●CUMULATIVE WEIGHTS
- ●NODE TYPES
- ●DATA LEVEL MASK
- ●AGGREGATE NODE INDICES
- ●SUCCESSOR TABLE
- ●SYSTEMS LABELS

1. The OUTLINE TABLE contains an element for each node in the model, sorted in increasing numerical sequence order. The value is an encoded representation of the node outline number supplied for a node when the model structure is created.

| | | | |
|---|---|---|---|
| System/Program: STRUCTURE | Name: LOADVARS | | |
| Diagram ID: 2.2 | Description: Read in the Variables | Page: 2 | of 3 |

**INPUT**

**PROCESS**

**OUTPUT**

**Extended Description**

2. The NODE LABELS contain descriptions (one per node in the same order as the outline table) of nodes that are supplied when the model structure is created.

3. SCORES is a numeric array which contains a set of values for each node of the structure. Each set of values consists of one number per system defined in the model.

4. WEIGHTS is a numeric vector containing the relative-importance values assigned to each node in the model structure. The elements must appear in the same order as the associated outline numbers. When a model structure is created, the vector is null or contains zeros.

5. For each element in the node outline table, there is an associated element in the CUMULATIVE WEIGHTS vector. The vector will contain the percentage of importance with respect to the entire model when all WEIGHTS have been entered.

6. The NODE TYPES are indicators of the type of calculation that is to be used in assessing SCORES and WEIGHTS.

7. The DATA LEVEL MASK indicates which nodes are at the data level (bottom level) versus the nodes that are aggregate or non-bottom-level nodes.

8. The AGGREGATE NODE INDICES contain the sequence number of elements in the model variables which correspond to only the aggregate nodes. An Aggregate node is a node which has one or more subsequent nodes contributing to it.

| | | | |
|---|---|---|---|
| System/Program: STRUCTURE | Name: LOADVARS | | |
| Diagram ID: 2.2 | Description Read in the Variables | Page: 3 | of 3 |

**INPUT**

**PROCESS**

**OUTPUT**

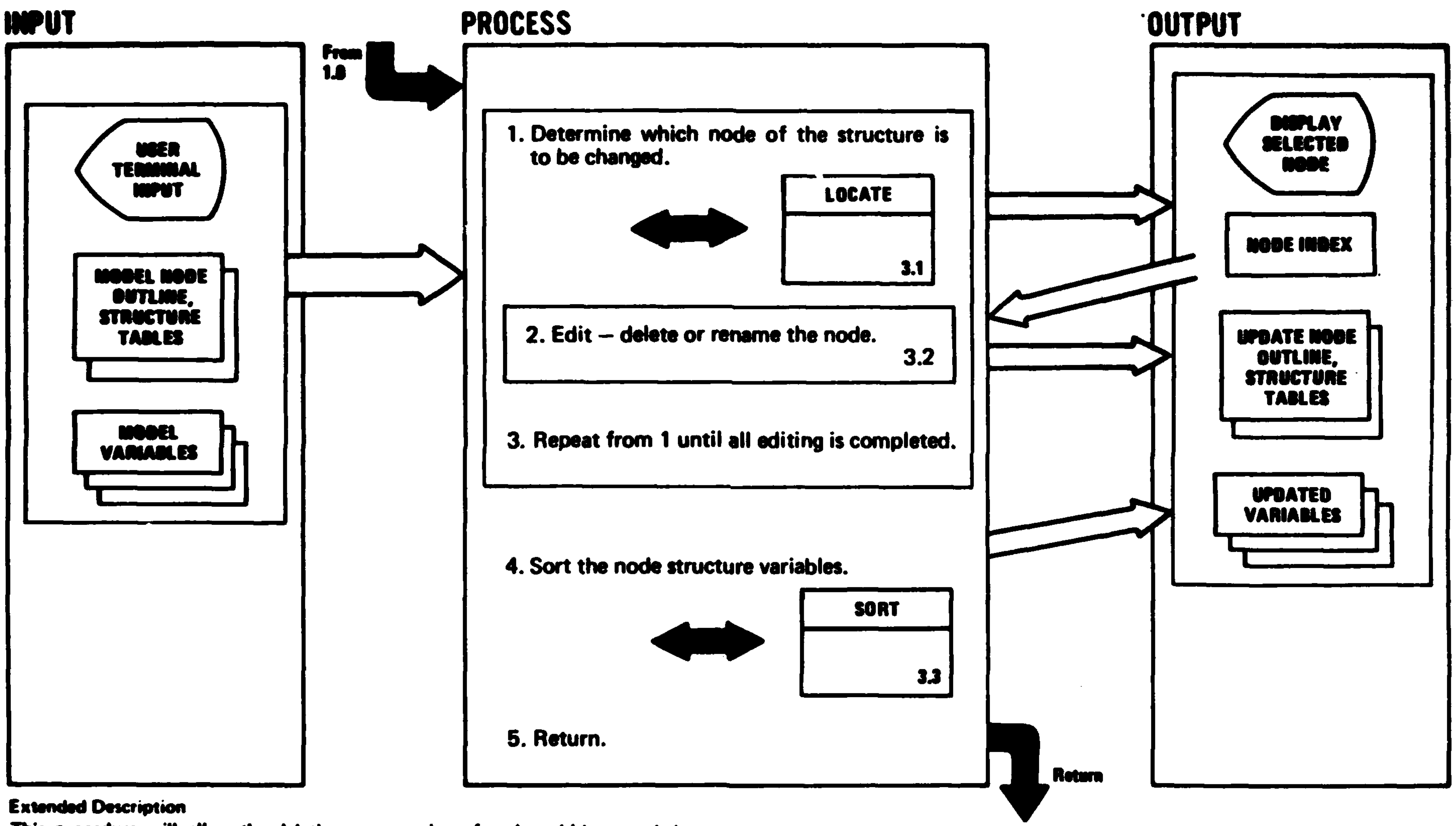
**Extended Description**

9. The SUCCESSOR TABLE is an array which contains, for each aggregate node, the set of indices of nodes which contribute to a node.

10. The SYSTEMS LABELS contain the user-specified character descriptions of the systems being evaluated.

System Program: STRUCTURE Name:

Diagram ID: 3.0 Description: Edit the Structure Page: of


INPUT
PROCESS
OUTPUT
USER TERMINAL INPUT
MODEL NODE OUTLINE, STRUCTURE TABLES
MODEL VARIABLES
From 1.0
1. Determine which node of the structure is to be changed.
LOCATE
3.1
2. Edit – delete or rename the node.
3.2
3. Repeat from 1 until all editing is completed.
4. Sort the node structure variables.
SORT
3.3
5. Return.
Return
DISPLAY SELECTED NODE
NODE INDEX
UPDATE NODE OUTLINE, STRUCTURE TABLES
UPDATED VARIABLES

Extended Description

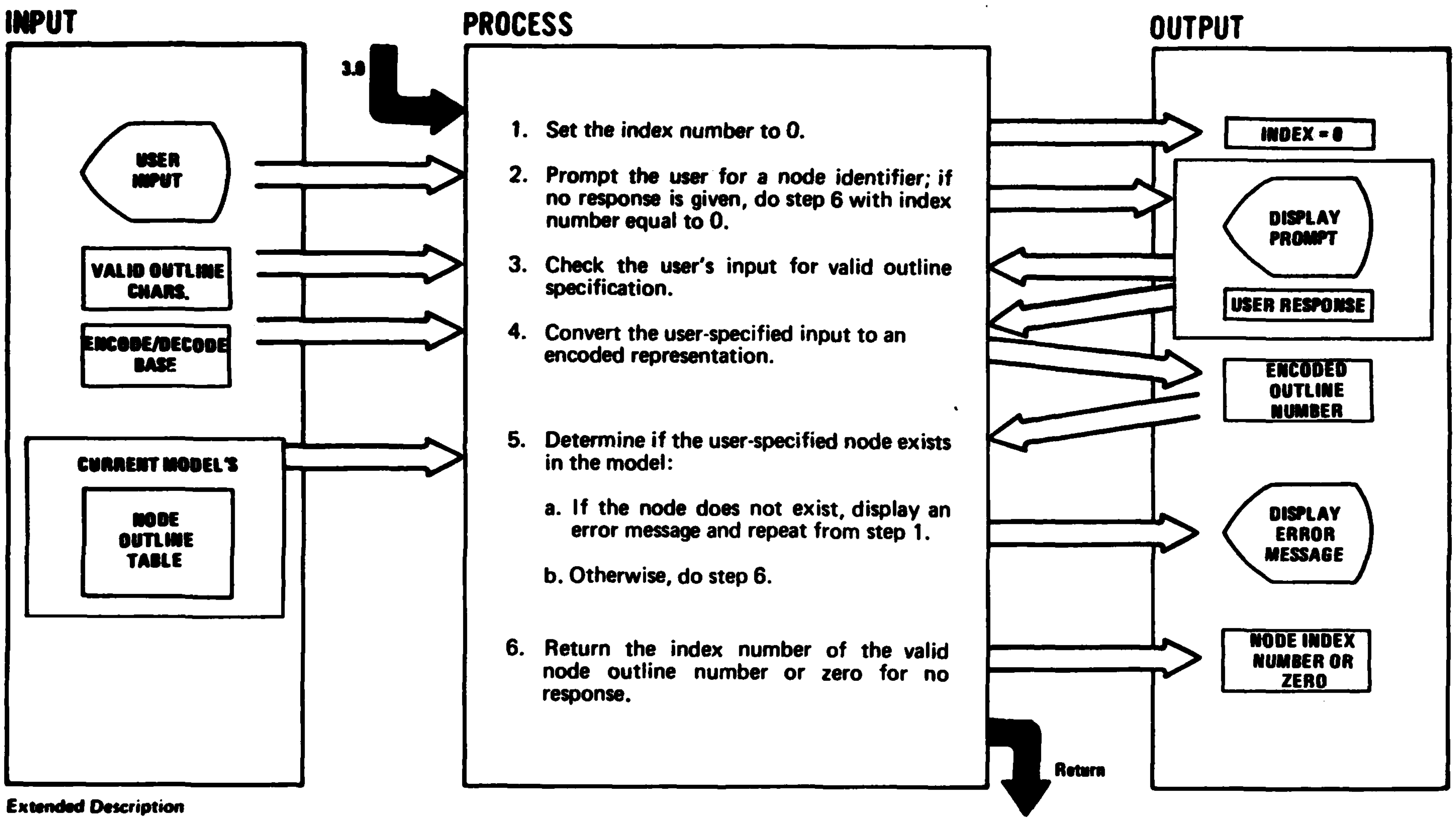
This procedure will allow the deletion or renaming of nodes within an existing structure and operates on a single node at a time. If a group or subtree of nodes is to be deleted, the user should select the "Prune a section" procedure described in diagram 8.0.

1. The user is prompted for a node identifier. This identifier corresponds to the manner in which the node was named when it was placed in the structure. The outline number is a shortened form of the node's identification. An associated index number is determined which is relative to the node outline and structure tables.

4. The node structure variables are reorganized so that associated nodes are always grouped together after the structure has been edited.

System/Program: STRUCTURE Name: LOCATE

Diagram ID: 3.1 Description Determine the Node Location Page: ___ of ___



1. Set the index number to 0.
2. Prompt the user for a node identifier; if no response is given, do step 6 with index number equal to 0.
3. Check the user's input for valid outline specification.
4. Convert the user-specified input to an encoded representation.
5. Determine if the user-specified node exists in the model:

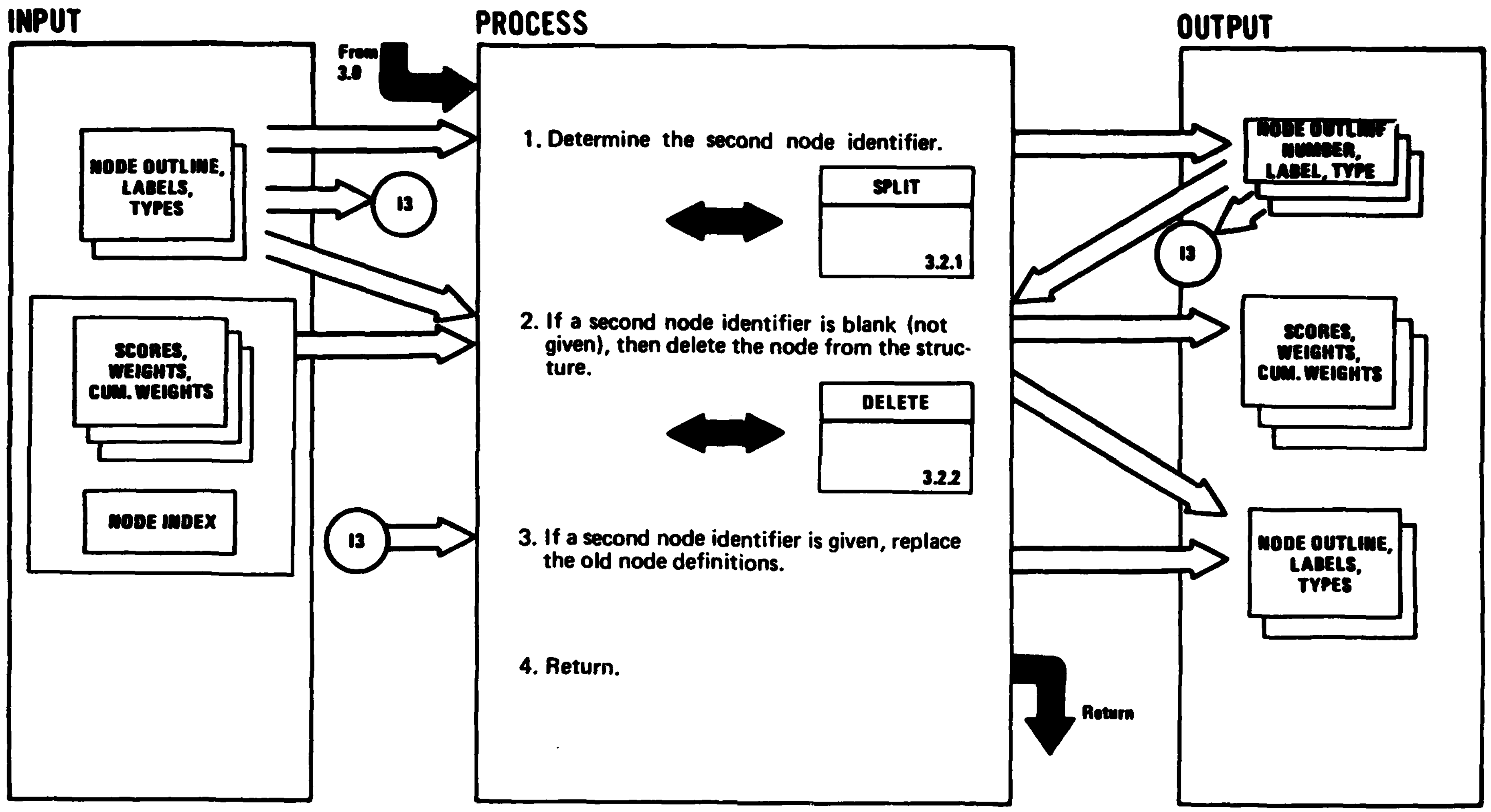
   a. If the node does not exist, display an error message and repeat from step 1.

   b. Otherwise, do step 6.
6. Return the index number of the valid node outline number or zero for no response.

**Extended Description**

**5. The existing outline table is searched for a matching encoded outline number. It is the index into this table of the matching outline number which is returned to the calling routine in step 6.**

System Program: STRUCTURE Name: 
Diagram ID: 3.2 Description: EDIT - Delete or Rename Page: of


INPUT
PROCESS
OUTPUT
From 3.0
NODE OUTLINE, LABELS, TYPES
SCORES, WEIGHTS, CUM. WEIGHTS
NODE INDEX
13
SPLIT
3.2.1
DELETE
3.2.2
Return
NODE OUTLINE NUMBER, LABEL, TYPE
SCORES, WEIGHTS, CUM. WEIGHTS
NODE OUTLINE, LABELS, TYPES

1. Determine the second node identifier.

2. If a second node identifier is blank (not given), then delete the node from the structure.

3. If a second node identifier is given, replace the old node definitions.

4. Return.

Extended Description

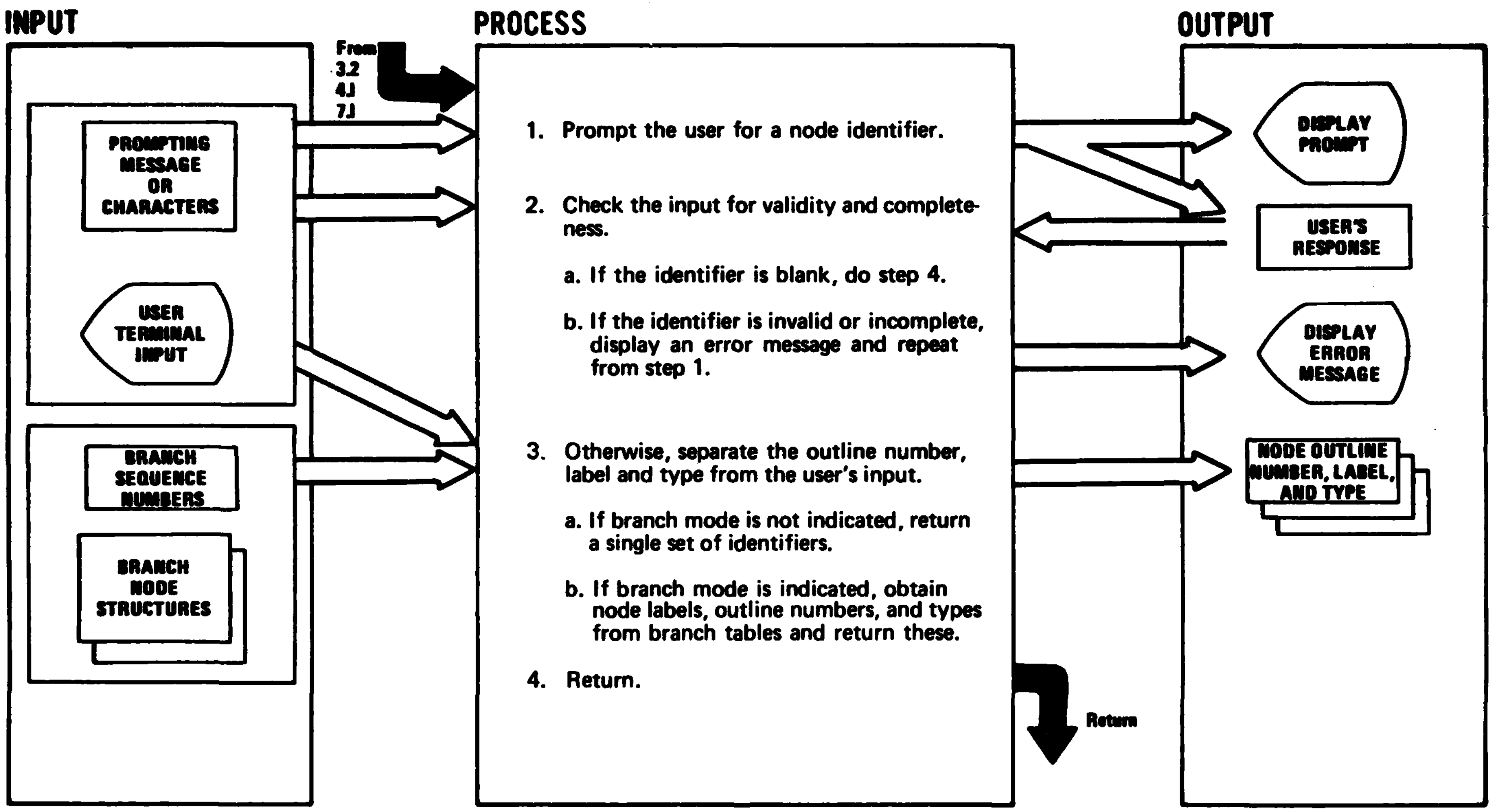
1. The user is prompted for all node identification information – the node outline number, the node label or name and its type. (See diagram 2.2 for a description of these items.)

2. A null entry or blank response from the user indicates that the node is to be deleted from the current structure.

3. Replace the outline number, the node label and type in the appropriate arrays with the new ones.

System Program: STRUCTURE Name: SPLIT

Diagram ID: 3.2.1 Description Investigate and Split Node Identifiers Page: ___ of ___



1. Prompt the user for a node identifier.
2. Check the input for validity and completeness.
   a. If the identifier is blank, do step 4.
   b. If the identifier is invalid or incomplete, display an error message and repeat from step 1.
3. Otherwise, separate the outline number, label and type from the user's input.
   a. If branch mode is not indicated, return a single set of identifiers.
   b. If branch mode is indicated, obtain node labels, outline numbers, and types from branch tables and return these.
4. Return.

**Extended Description**

1. The user is required to input the identifying information for a particular node in either an existing structure or one that is currently being defined.

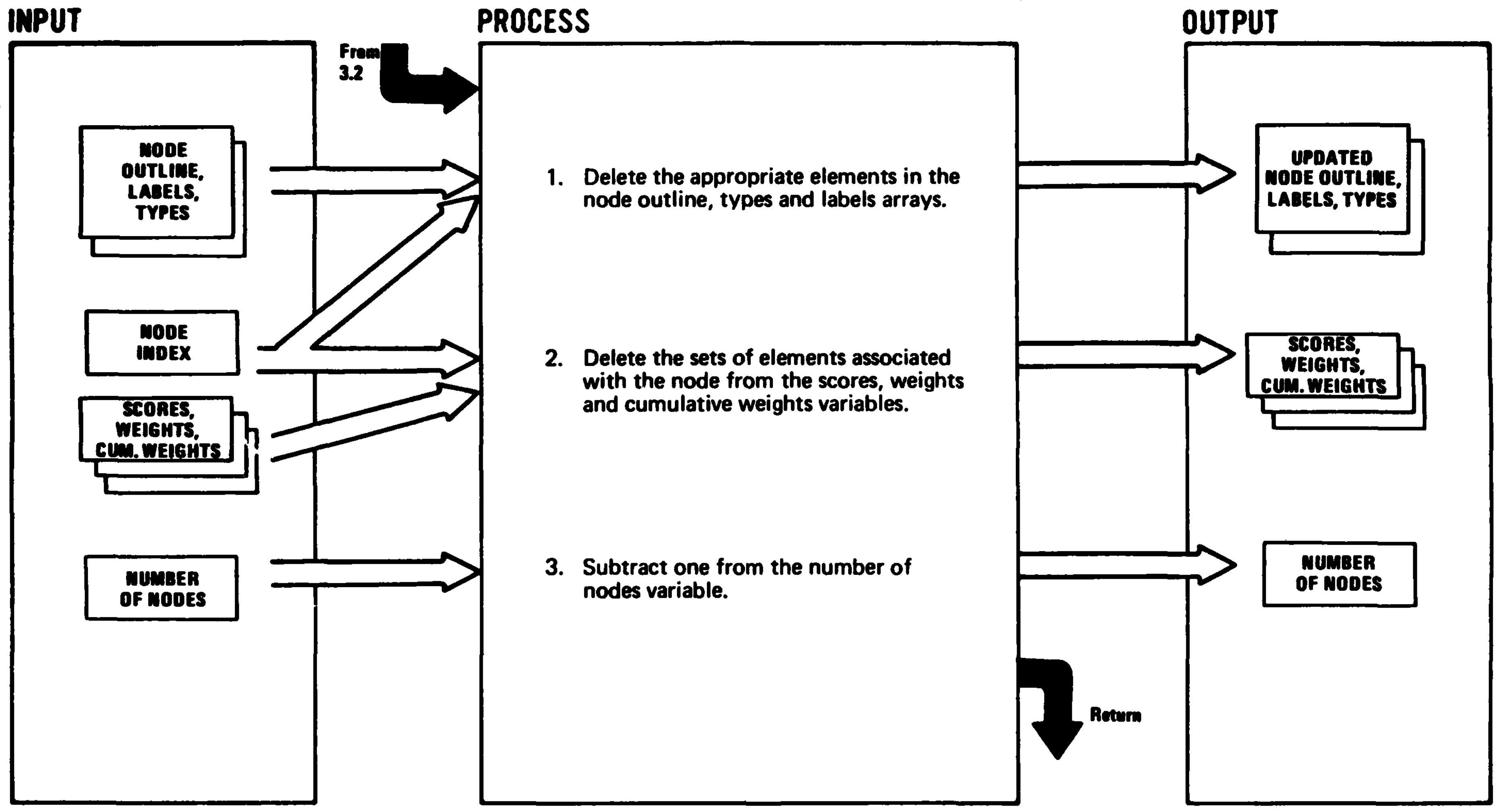
2. Proper node identification consists of an outline sequence number which has a hierarchical relationship to other nodes in the structure, a label or descriptive name, and a node "type" indicator. The three variables are usually entered with commas or some other punctuation separating each one from the other.

A special character, such as an asterisk (*) or pound sign (#), should be used to designate that a group or subtree is being specified. The special character would be the first in the input line of the user's response.

3. The outline number – numerically encoded to a sufficiently large number, the label, and type are returned as separate variables.

If a branch or subtree is being specified, the appropriate node labels, outline numbers and types are obtained from the branch structure tables. A group of encoded outline numbers, a group of labels and the group types are all returned to the calling routine. The new outline numbers have been encoded again to agree with the node after which the branch or subtree is being placed in hierarchical fashion.

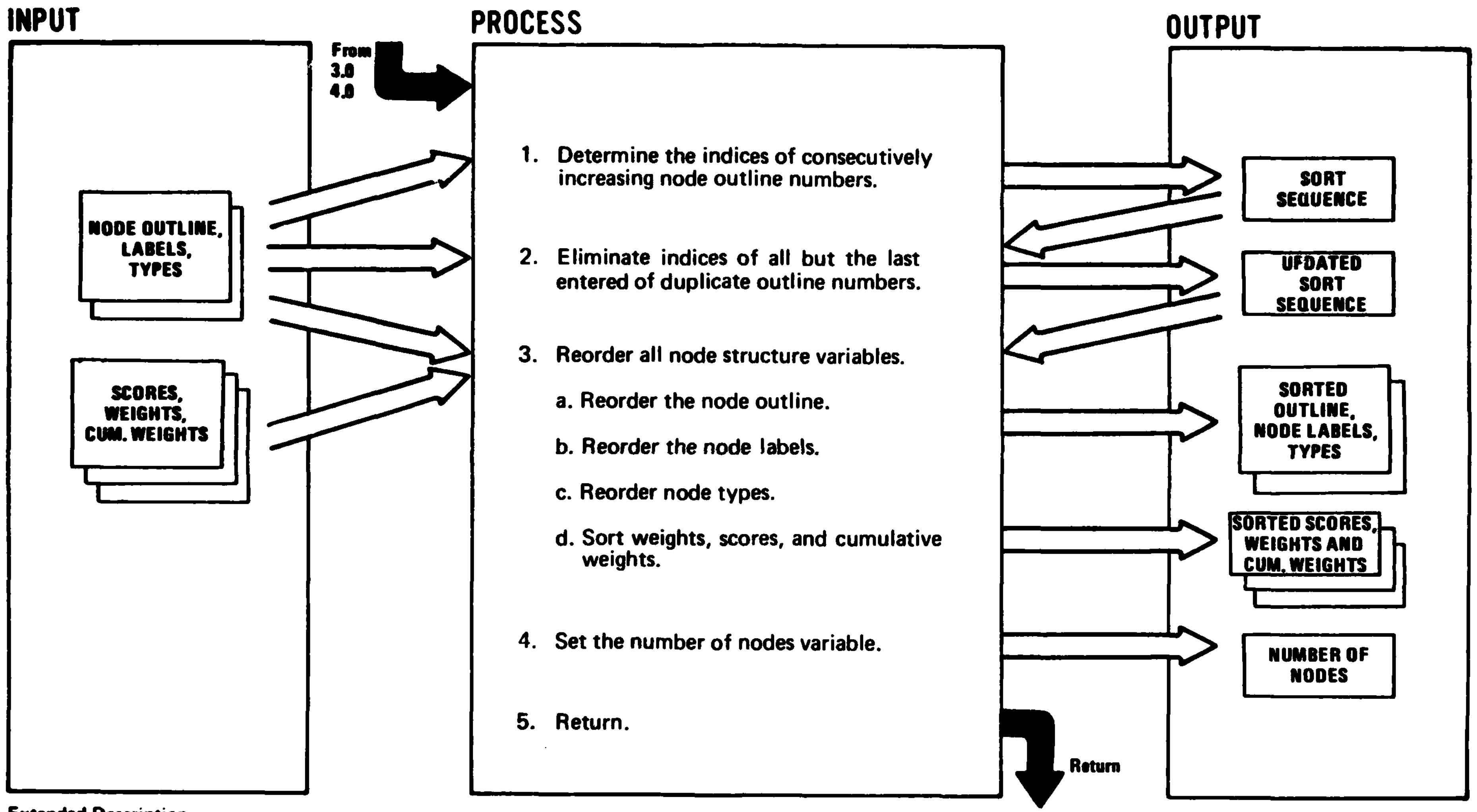
System/Program: STRUCTURE Name:
Diagram ID: 3.2.2 Description Delete a Node Page: of


INPUT
NODE OUTLINE, LABELS, TYPES
NODE INDEX
SCORES, WEIGHTS, CUM. WEIGHTS
NUMBER OF NODES
From 3.2
PROCESS
Return
OUTPUT
UPDATED NODE OUTLINE, LABELS, TYPES
SCORES, WEIGHTS, CUM. WEIGHTS
NUMBER OF NODES

1. Delete the appropriate elements in the node outline, types and labels arrays.
2. Delete the sets of elements associated with the node from the scores, weights and cumulative weights variables.
3. Subtract one from the number of nodes variable.

System Program: STRUCTURE Name: SORT

Diagram ID: 3.3 Description SORT the Node Structure Variables Page: ___ of ___


INPUT
NODE OUTLINE, LABELS, TYPES
SCORES, WEIGHTS, CUM. WEIGHTS
From 3.0 4.0
PROCESS
OUTPUT
SORT SEQUENCE
UFDATED SORT SEQUENCE
SORTED OUTLINE, NODE LABELS, TYPES
SORTED SCORES, WEIGHTS AND CUM. WEIGHTS
NUMBER OF NODES
Return

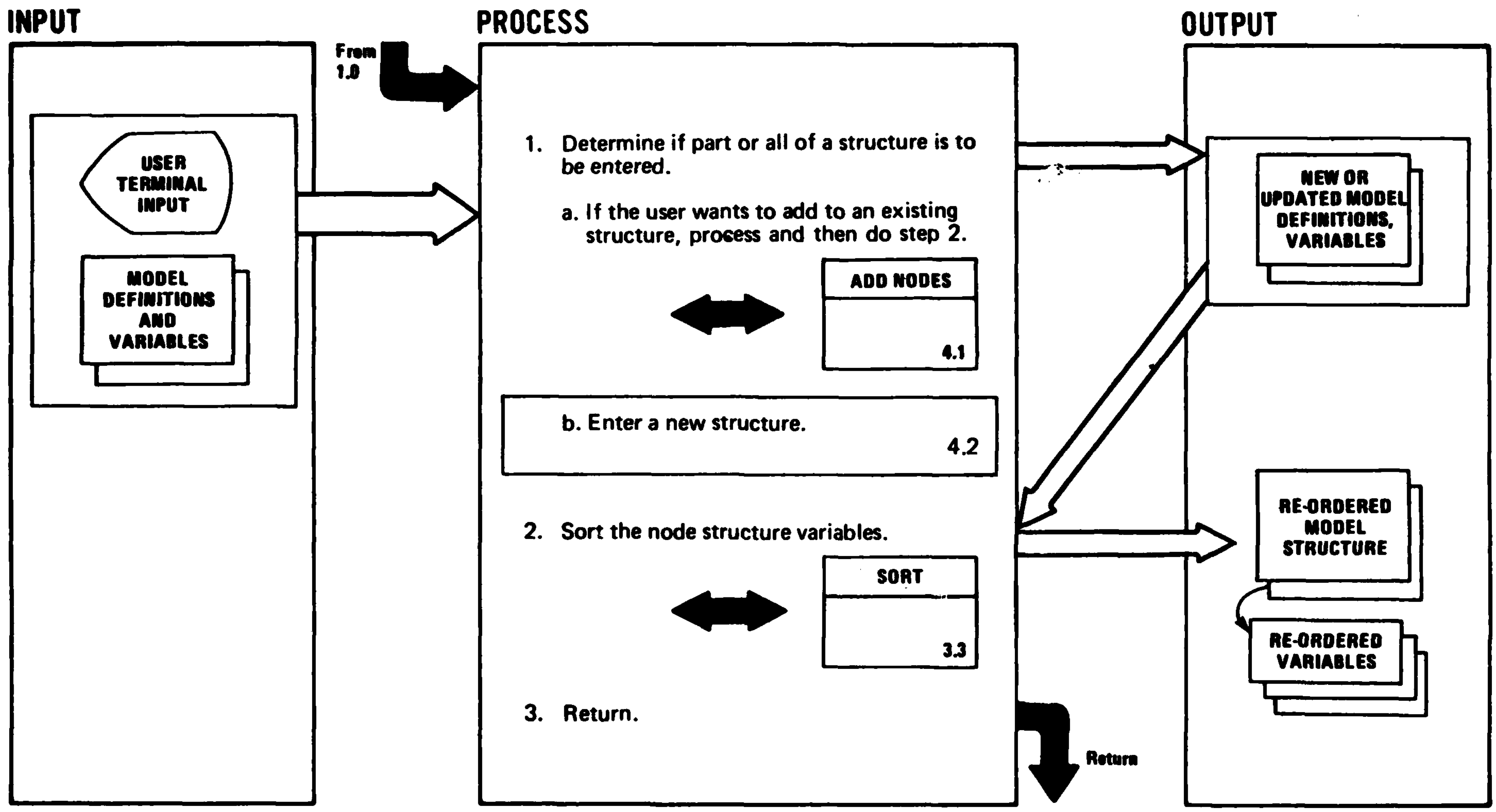
1. Determine the indices of consecutively increasing node outline numbers.
2. Eliminate indices of all but the last entered of duplicate outline numbers.
3. Reorder all node structure variables.
   a. Reorder the node outline.
   b. Reorder the node labels.
   c. Reorder node types.
   d. Sort weights, scores, and cumulative weights.
4. Set the number of nodes variable.
5. Return.

**Extended Description**

1. The relative indices or locations in the numerically encoded set of outline numbers in increasing value are determined. These indices constitute the sort sequence and will be used to rearrange the structure variables.

System/Program: STRUCTURE Name:

Diagram ID: 4.0 Description Create or Add to a Structure Page: of


INPUT
PROCESS
OUTPUT
USER TERMINAL INPUT
MODEL DEFINITIONS AND VARIABLES
From 1.0
ADD NODES
4.1
SORT
3.3
NEW OR UPDATED MODEL DEFINITIONS, VARIABLES
RE-ORDERED MODEL STRUCTURE
RE-ORDERED VARIABLES
Return

1. Determine if part or all of a structure is to be entered.

   a. If the user wants to add to an existing structure, process and then do step 2.

   b. Enter a new structure. 4.2

2. Sort the node structure variables.

3. Return.

Extended Description

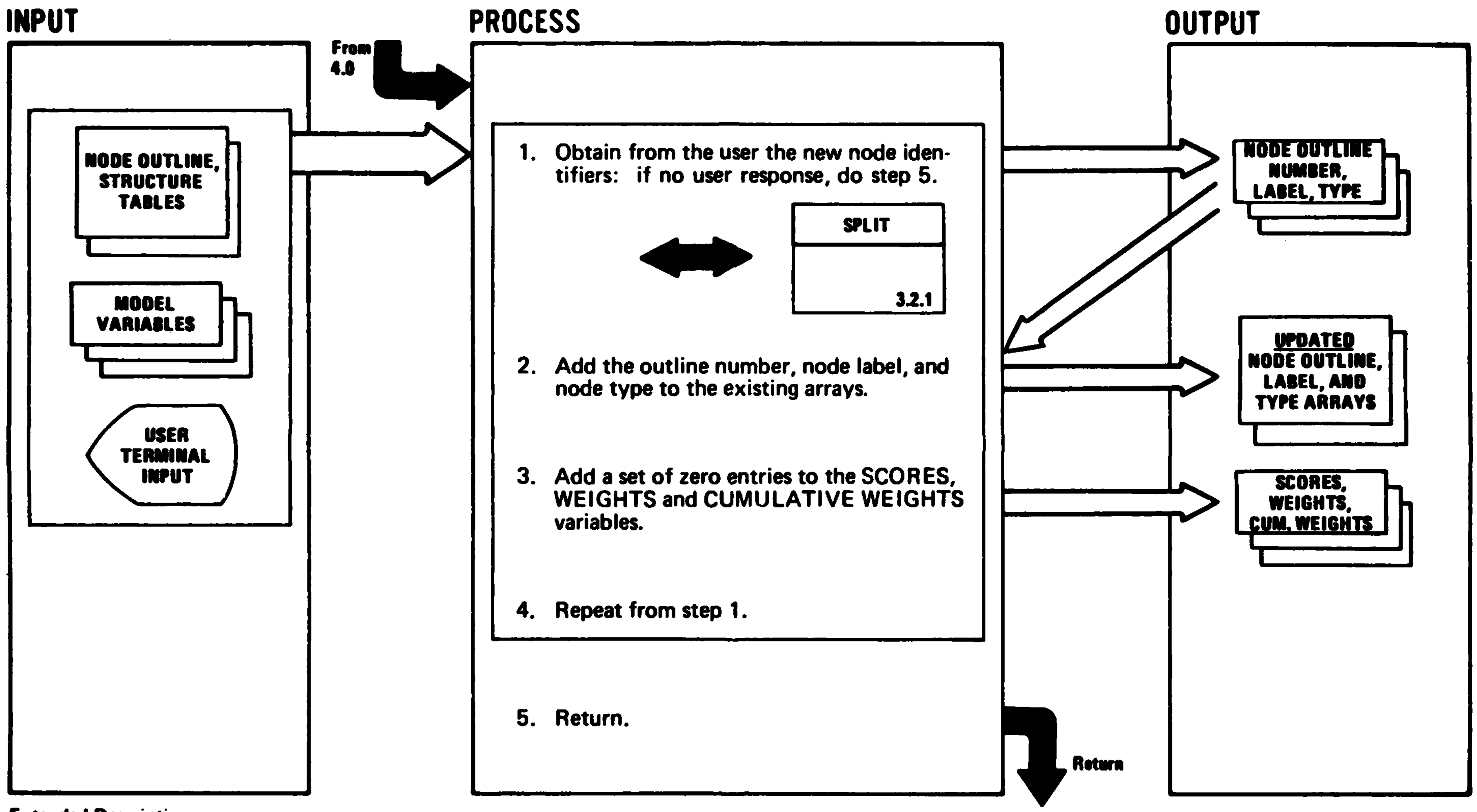
1. Request a "yes" or "no" response directly from the user to determine whether a new structure is to be entered or nodes are to be added to an existing structure.

b. If a new structure is entered, all currently defined variables of the old structures are deleted.

2. An explanation of the sorting function is given in diagram 3.3 of the STRUCTURE System Specifications.

System/Program: STRUCTURE Name: ADDNODES

Diagram ID: 4.1 Description Add Nodes to a Structure Page: ____ of ____


INPUT
NODE OUTLINE, STRUCTURE TABLES
MODEL VARIABLES
USER TERMINAL INPUT
From 4.0
PROCESS
SPLIT
3.2.1
Return
OUTPUT
NODE OUTLINE NUMBER, LABEL, TYPE
UPDATED NODE OUTLINE, LABEL, AND TYPE ARRAYS
SCORES, WEIGHTS, CUM. WEIGHTS

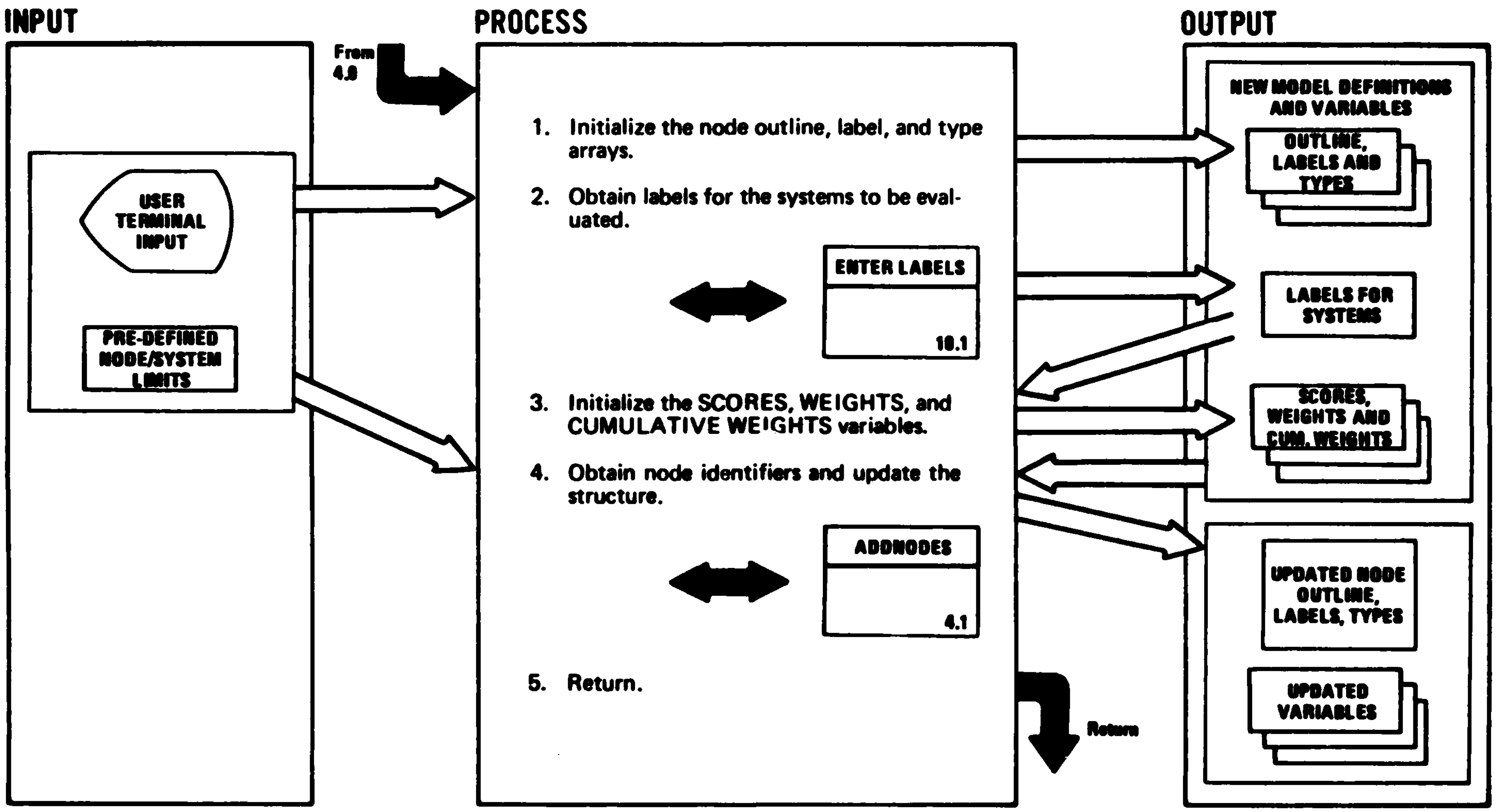
1. Obtain from the user the new node identifiers: if no user response, do step 5.
2. Add the outline number, node label, and node type to the existing arrays.
3. Add a set of zero entries to the SCORES, WEIGHTS and CUMULATIVE WEIGHTS variables.
4. Repeat from step 1.
5. Return.

**Extended Description**

**2 - 3. Additions to previously initialized or existing variables are accomplished by extending the arrays such that the corresponding orders of associated labels, scores, types, weights and decoded outline numbers are the same.**

System/Program: STRUCTURE Name:

Diagram ID: 4.2 Description: Enter a New Structure Page: of


INPUT
PROCESS
OUTPUT
USER TERMINAL INPUT
PRE-DEFINED NODE/SYSTEM LIMITS
From 4.8
ENTER LABELS
10.1
ADDNODES
4.1
Return
NEW MODEL DEFINITIONS AND VARIABLES
OUTLINE, LABELS AND TYPES
LABELS FOR SYSTEMS
SCORES, WEIGHTS AND CUM. WEIGHTS
UPDATED NODE OUTLINE, LABELS, TYPES
UPDATED VARIABLES

1. Initialize the node outline, label, and type arrays.
2. Obtain labels for the systems to be evaluated.
3. Initialize the SCORES, WEIGHTS, and CUMULATIVE WEIGHTS variables.
4. Obtain node identifiers and update the structure.
5. Return.

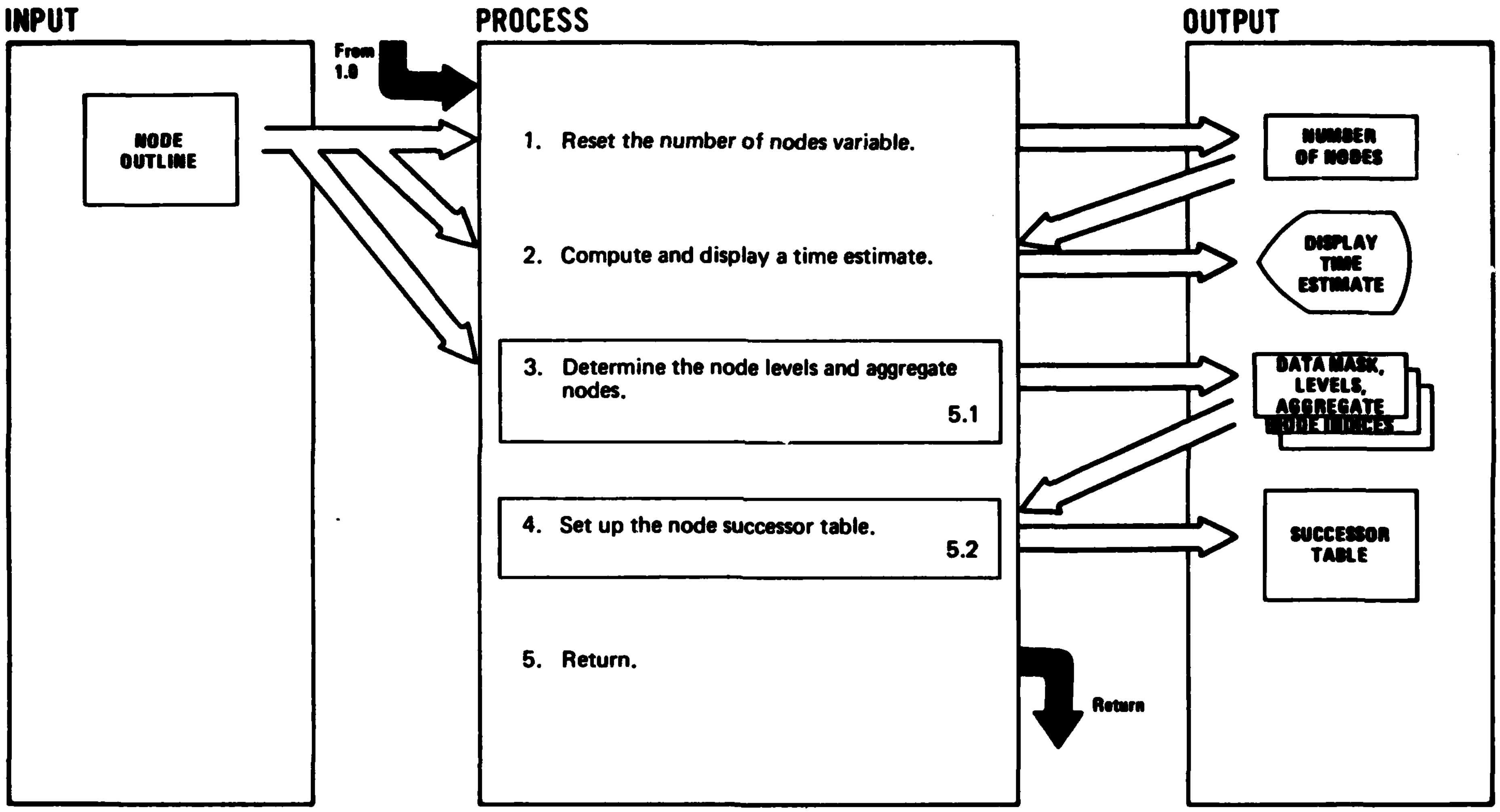
**Extended Description**

1. Initialization is caused by establishing null or blank vectors for the specified variables.

2. Labels for the systems to be evaluated are obtained from later storage and for the determination of the length of any set of SCORES.

4. The user is prompted for input which will be used to define a hierarchical tree structure described by outline numbers, labels and types of nodes within the structure.

| | |
|---|---|
| System Program: STRUCTURE | Name: |
| Diagram ID: 5.0 | Description: Develop the Structure — Page: of |



1. Reset the number of nodes variable.
2. Compute and display a time estimate.
3. Determine the node levels and aggregate nodes. 5.1
4. Set up the node successor table. 5.2
5. Return.

Extended Description

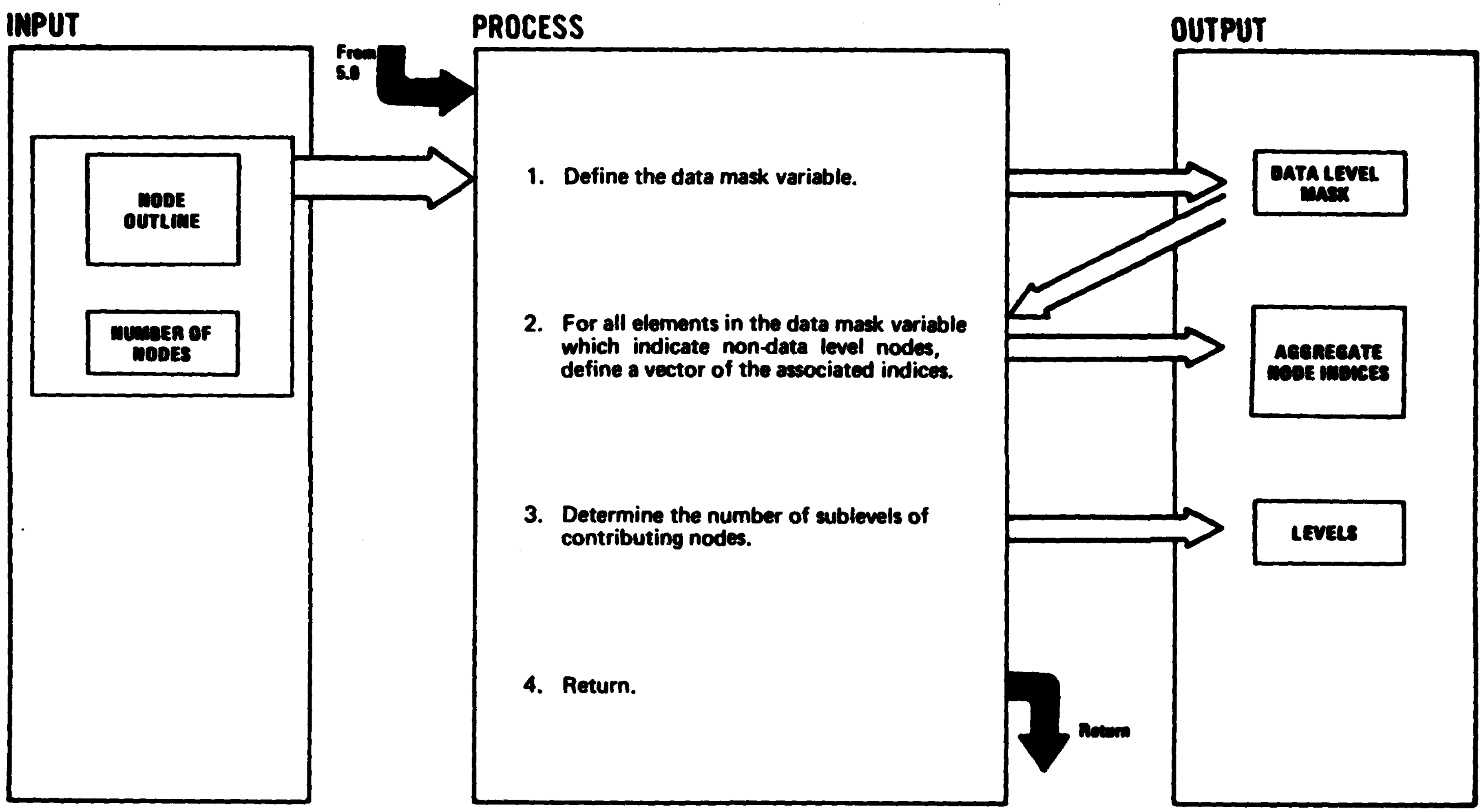
1. The number of nodes is equal to the number of entries in the outline array.

2. A rough estimate of the amount of time required to perform the developing operation may be displayed. The estimate is derived from the number of nodes in the model.

3. The data level mask indicates which nodes in the model are at the data level and which nodes are aggregate nodes. The aggregate node indices are indices into the node outline of nodes which are not at the data level. The LEVELS variable shows how far away a particular node is from the lowest level.

4. The successor table provides a set of contributing node indices for each aggregate node in the same order as aggregate node appearance in the outline.

System/Program: STRUCTURE Name:
Diagram ID: 5.1 Description Determine the Node Levels Page: of



1. Define the data mask variable.
2. For all elements in the data mask variable which indicate non-data level nodes, define a vector of the associated indices.
3. Determine the number of sublevels of contributing nodes.
4. Return.

**Extended Description**

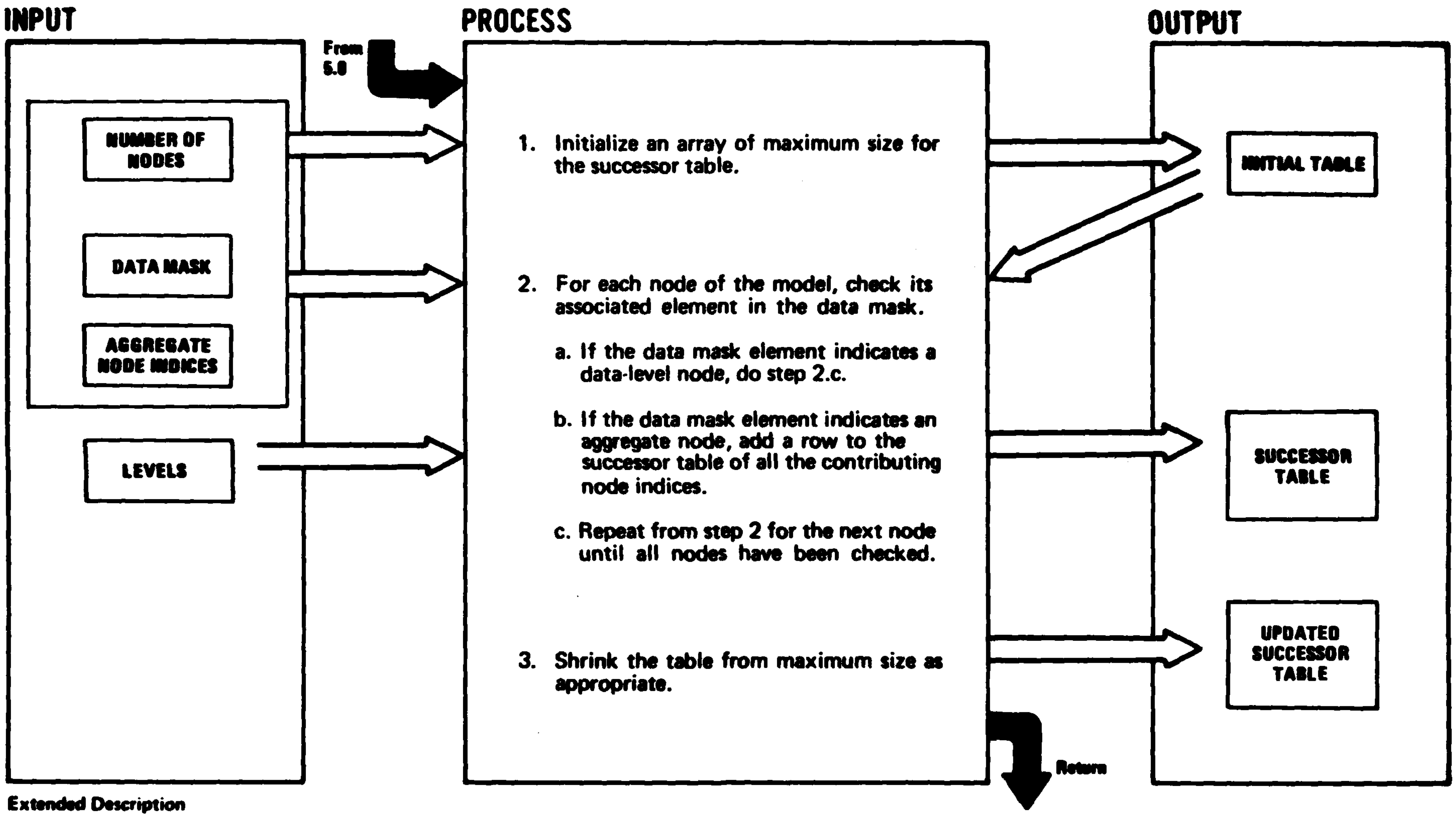
1. For each node in the model outline, an element is placed in a vector to indicate that node is a data level node or that it is an aggregate node having other contributing nodes.

The indicator may be 0 for data level and 1 for the aggregate level or vice versa.

2. The data level mask indicator setting for each node in the outline is used to determine the aggregate node indices – indices into the node outline.

3. The farthest element or data level node from the topmost node is determined. The topmost node is assigned the number of levels between it and the data level farthest away (the depth of the path with the most sub-level tree branches). All other nodes are assigned a value equal to the top-level's minus its distance (number of levels) from the top.

System/Program: STRUCTURE Name:
Diagram ID: 5.2 Description Set up the Node Successor Table Page: of



1. Initialize an array of maximum size for the successor table.

2. For each node of the model, check its associated element in the data mask.

   a. If the data mask element indicates a data-level node, do step 2.c.

   b. If the data mask element indicates an aggregate node, add a row to the successor table of all the contributing node indices.

   c. Repeat from step 2 for the next node until all nodes have been checked.

3. Shrink the table from maximum size as appropriate.

**Extended Description**

1. The maximum size table is prescribed by the number of aggregate nodes and the predefined limit to the number of contributing nodes on any single level.

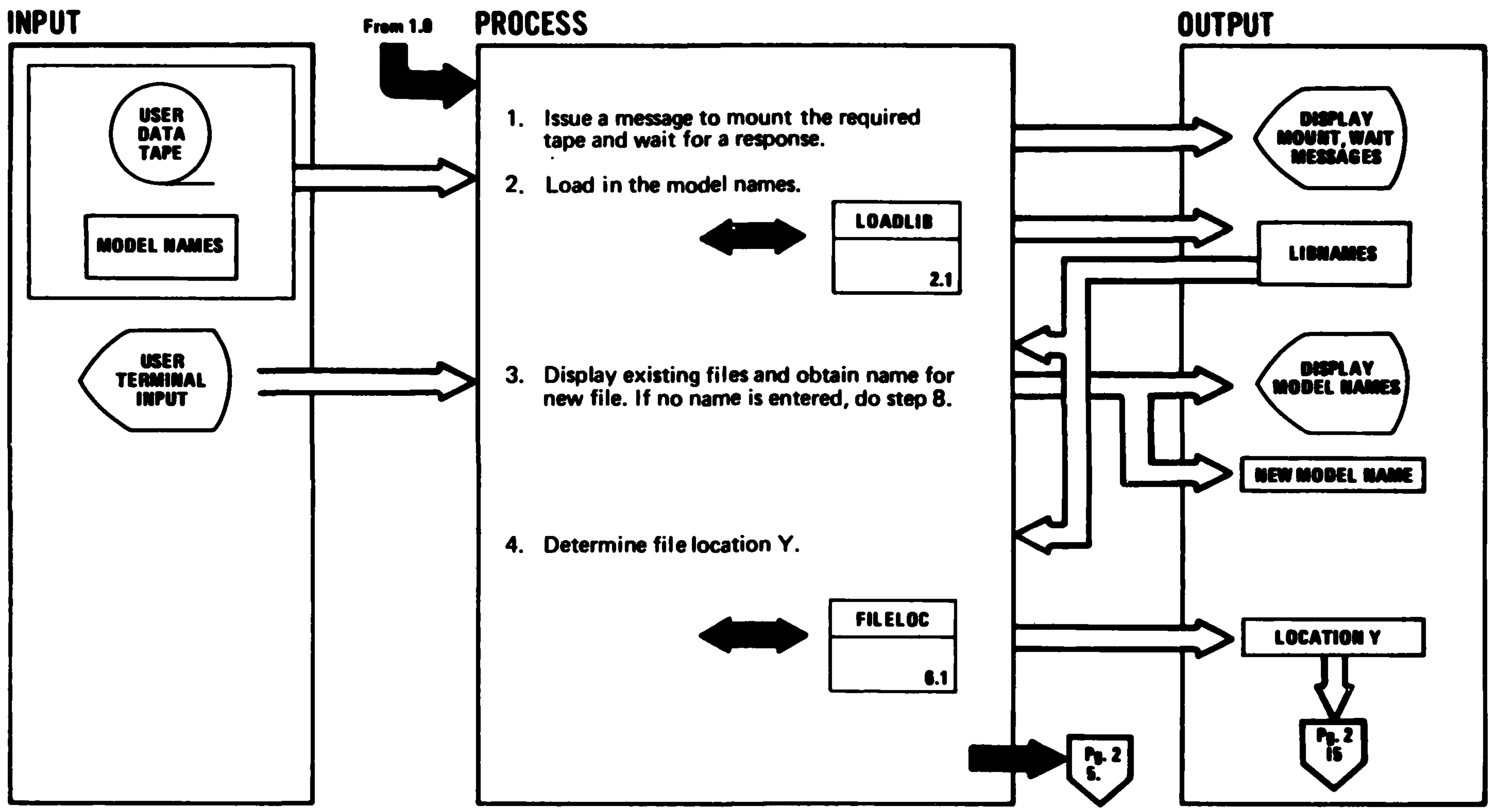
2. This procedure steps through the data mask variable in sequential order: the contributing nodes of the topmost aggregate node will be added to the successor table first.

2.b. If the nodes' associated data mask element indicates an aggregate node, then the contributing nodes are all the nodes which follow in sequential order that have an associated LEVELS number that is less than the selected nodes LEVELS numbers, provided these nodes occur before any node with equal or higher LEVELS number.

3. Since the number of elements in any set of contributing nodes may be less than the predefined limit, the number of columns (or characters) in the table may be diminished.

System/Program: STRUCTURE Name:

Diagram ID: 6.0 Description Save the Current Model Page: 1 of 2



1. Issue a message to mount the required tape and wait for a response.
2. Load in the model names.
3. Display existing files and obtain name for new file. If no name is entered, do step 8.
4. Determine file location Y.

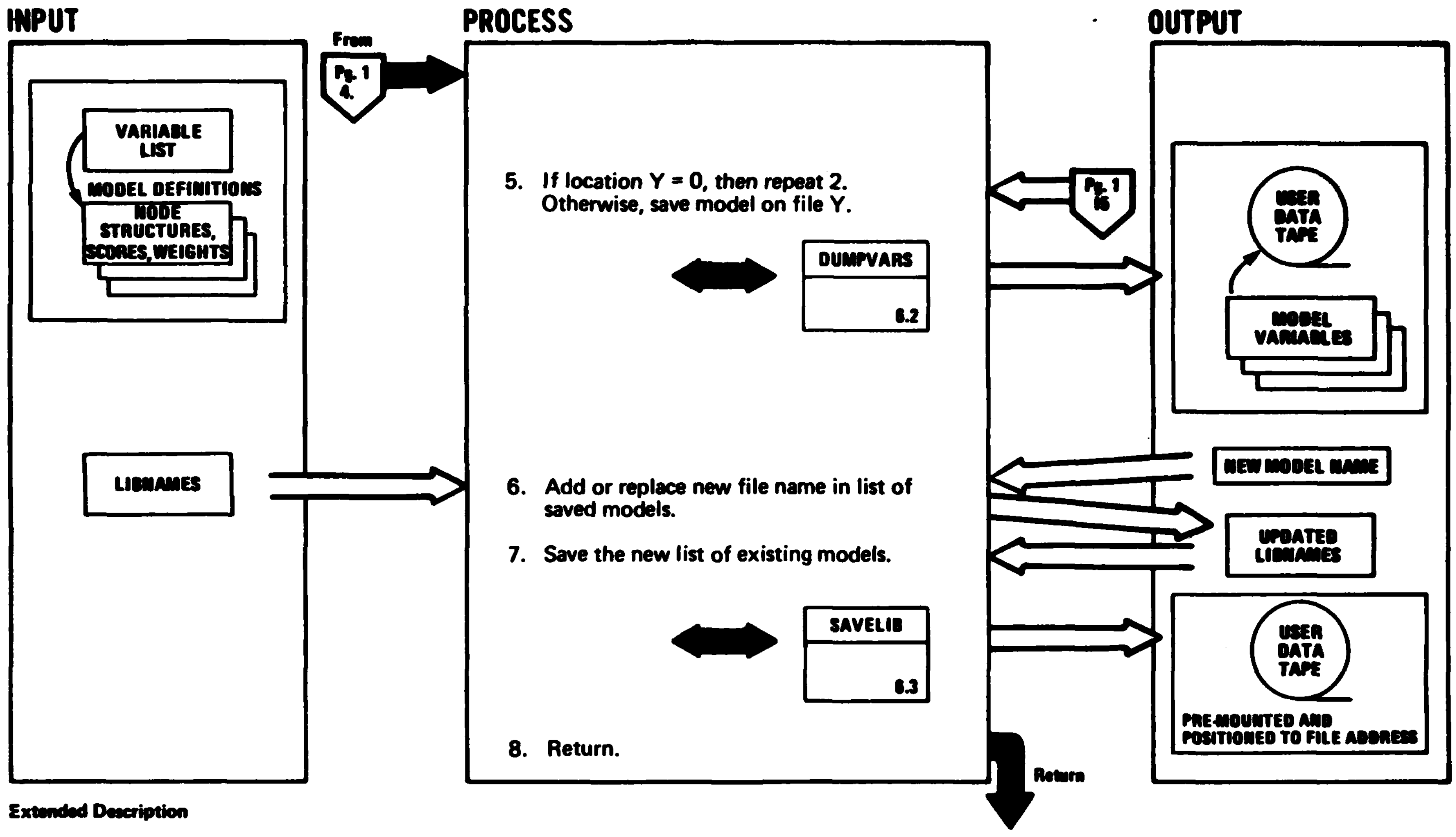
**Extended Description**

1. The computer program prompts for an indication that the desired storage file/device has been selected and placed online. Any response from the keyboard causes processing to resume.

4. The existing file structure and the amount of available space on the data tape are checked along with the user specification to determine where the model variables are to be stored.

System/Program: STRUCTURE Name:

Diagram ID: 6.0 Description Save the Current Model Page: 2 of 2



5. If location Y = 0, then repeat 2. Otherwise, save model on file Y.

6. Add or replace new file name in list of saved models.

7. Save the new list of existing models.

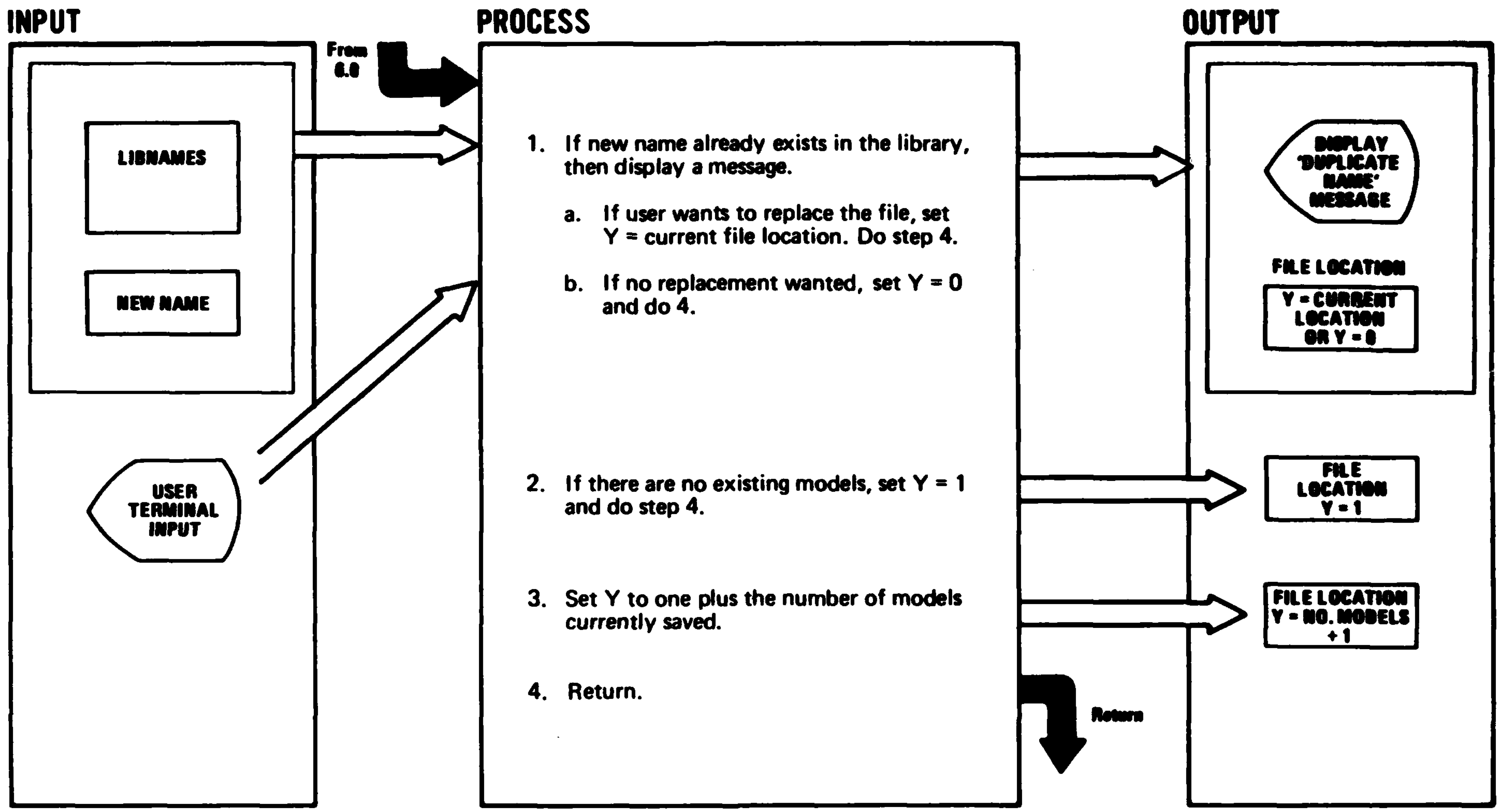
8. Return.

**Extended Description**

6. The library name list is updated to include the new file. The new model name's position in the LIBNAMES array must be the same relative position to other models stored on the device.

System/Program: STRUCTURE Name: FILELOC

Diagram ID: 6.1 Description Find File Location Page: ___ of ___



1. If new name already exists in the library, then display a message.

   a. If user wants to replace the file, set Y = current file location. Do step 4.

   b. If no replacement wanted, set Y = 0 and do 4.

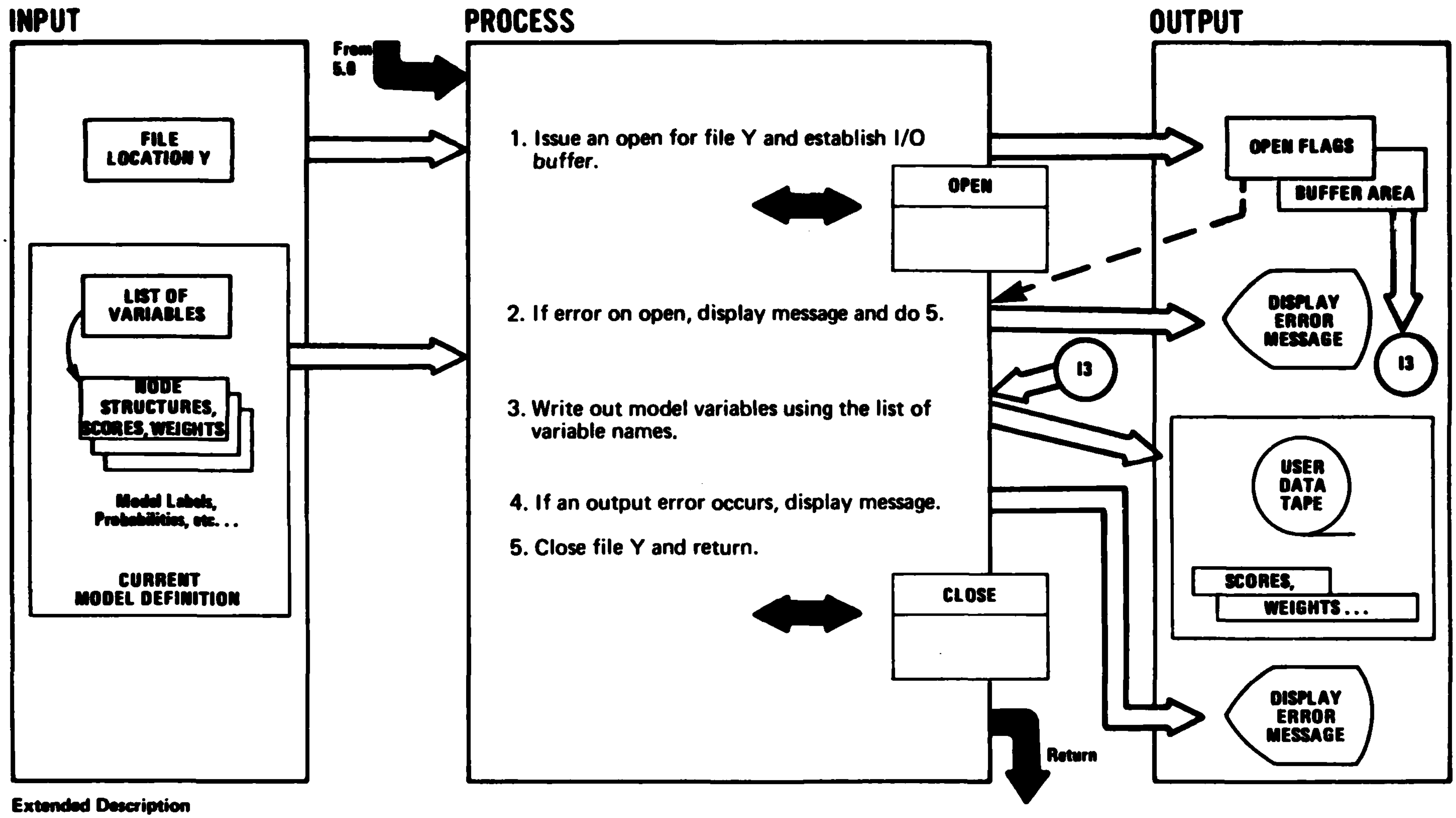
2. If there are no existing models, set Y = 1 and do step 4.

3. Set Y to one plus the number of models currently saved.

4. Return.

System/Program: STRUCTURE Name: DUMPVARS

Diagram ID: 6.2 Description Save the Model Variables Page: ___ of ___



1. Issue an open for file Y and establish I/O buffer.
2. If error on open, display message and do 5.
3. Write out model variables using the list of variable names.
4. If an output error occurs, display message.
5. Close file Y and return.

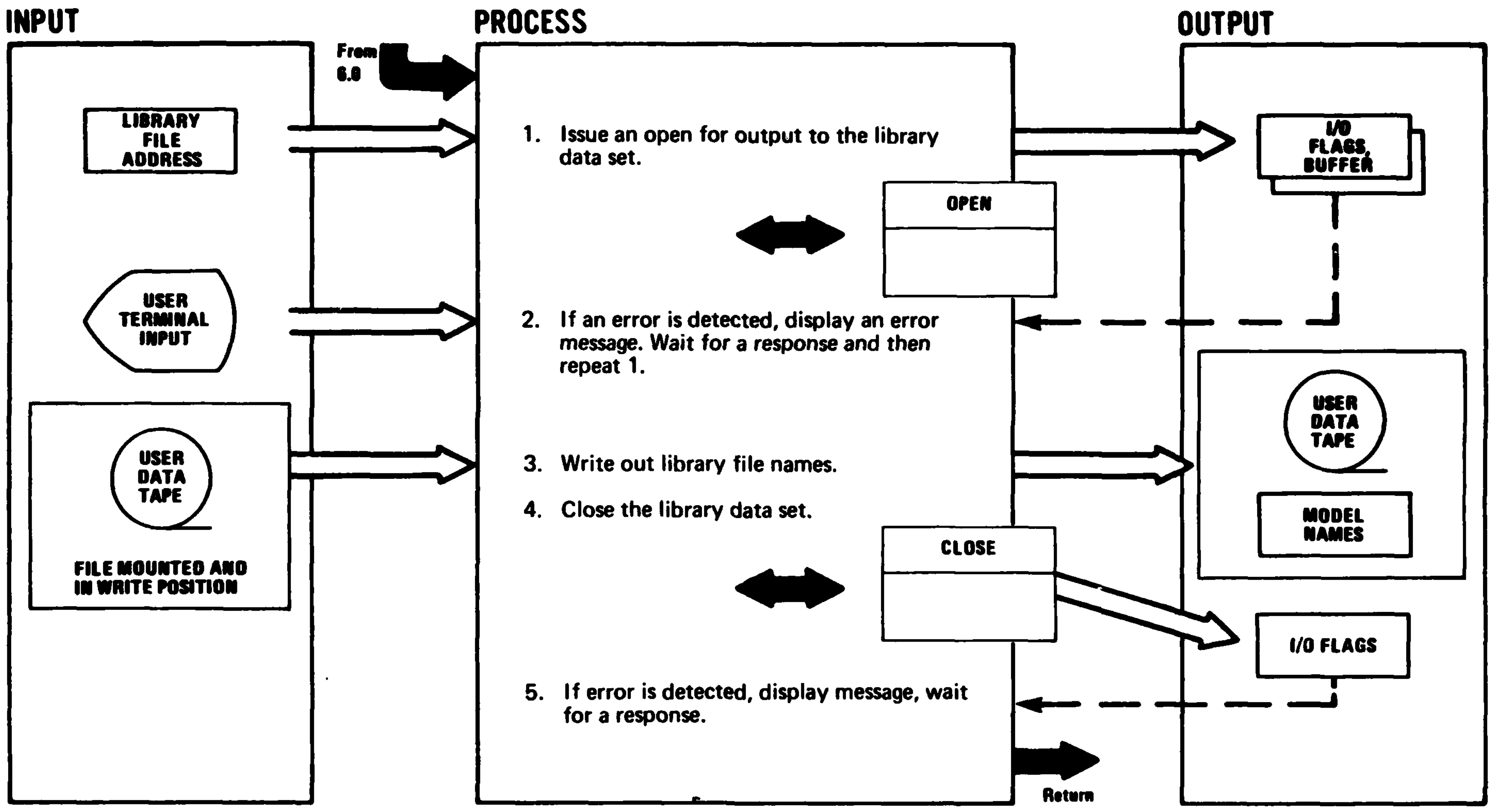
**Extended Description**

1. The file location Y is used to determine an exact storage position on the selected device.

3. The list of variable names is identical to the list of names used to Load a Model (see diagram 2.2)

System/Program: STRUCTURE Name: SAVELIB

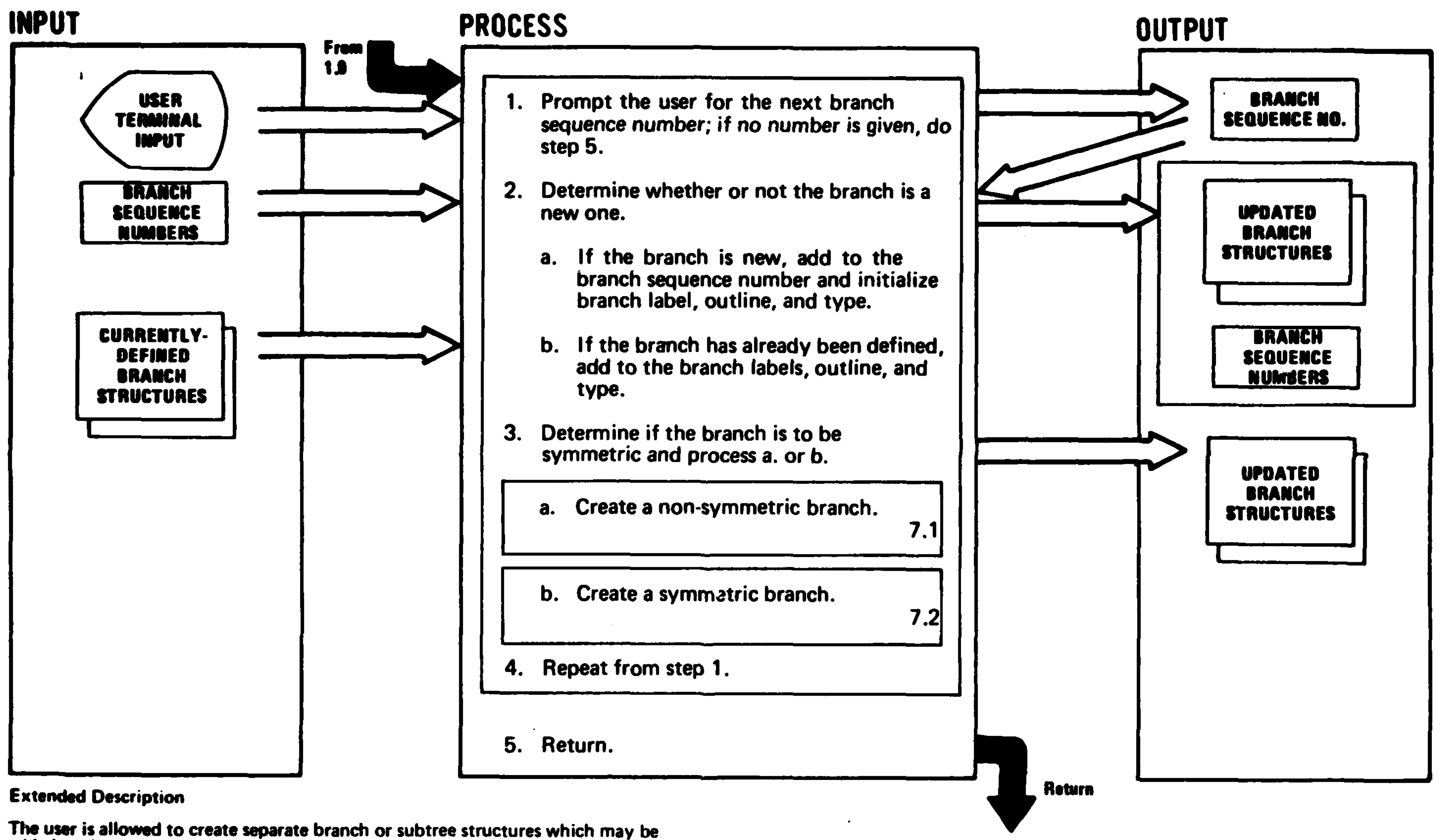
Diagram ID: 6.3 Description: Save the Library Page: ___ of ___



1. Issue an open for output to the library data set.
2. If an error is detected, display an error message. Wait for a response and then repeat 1.
3. Write out library file names.
4. Close the library data set.
5. If error is detected, display message, wait for a response.

System/Program: STRUCTURE Name: ______

Diagram ID: 7.0 Description Create a Branch to the Structure Page: ___ of ___



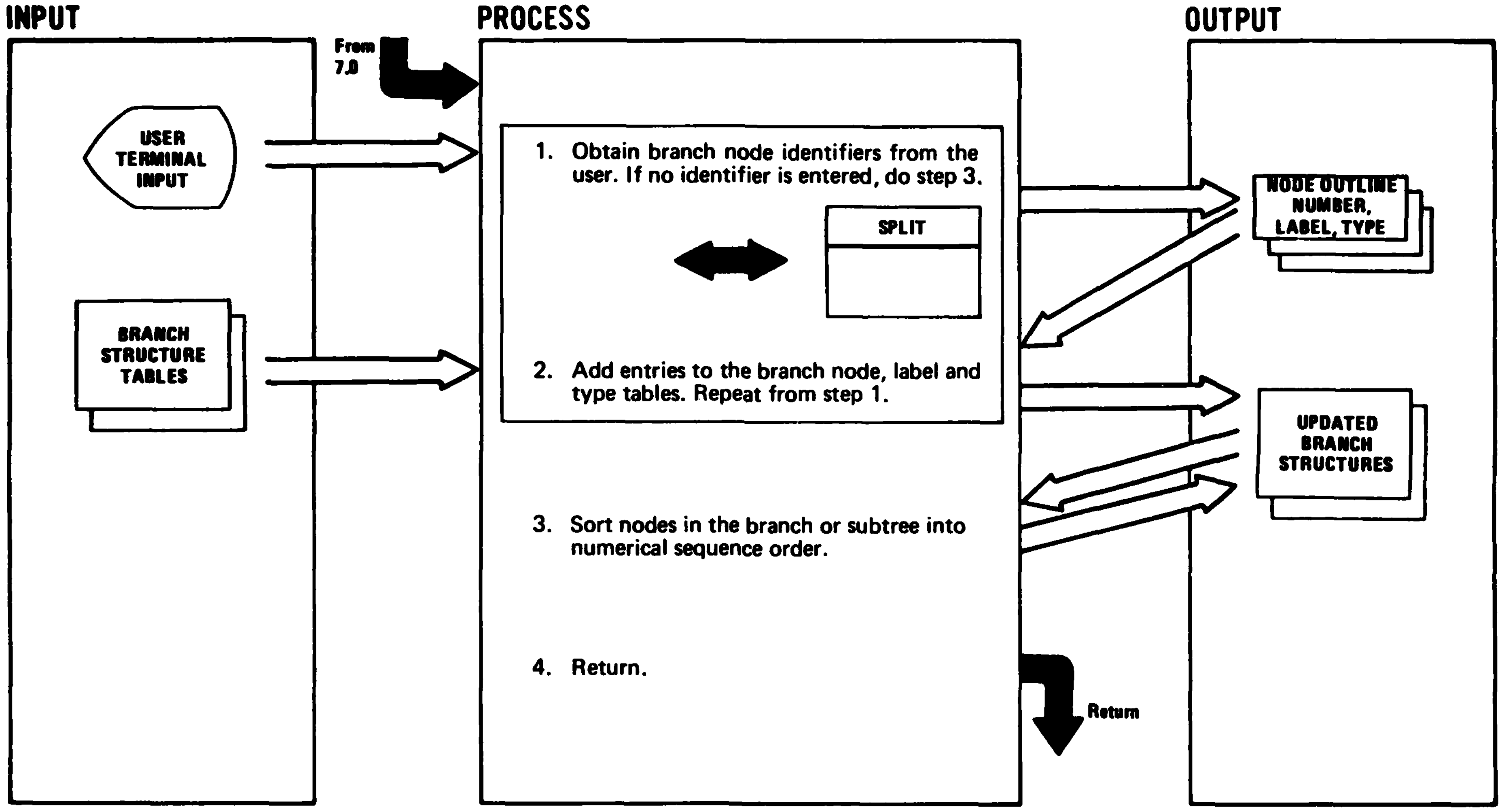
1. Prompt the user for the next branch sequence number; if no number is given, do step 5.
2. Determine whether or not the branch is a new one.
   a. If the branch is new, add to the branch sequence number and initialize branch label, outline, and type.
   b. If the branch has already been defined, add to the branch labels, outline, and type.
3. Determine if the branch is to be symmetric and process a. or b.
   a. Create a non-symmetric branch. 7.1
   b. Create a symmetric branch. 7.2
4. Repeat from step 1.
5. Return.

**Extended Description**

The user is allowed to create separate branch or subtree structures which may be added to the model structure under the "create a structure" process option.

System/Program: STRUCTURE Name:

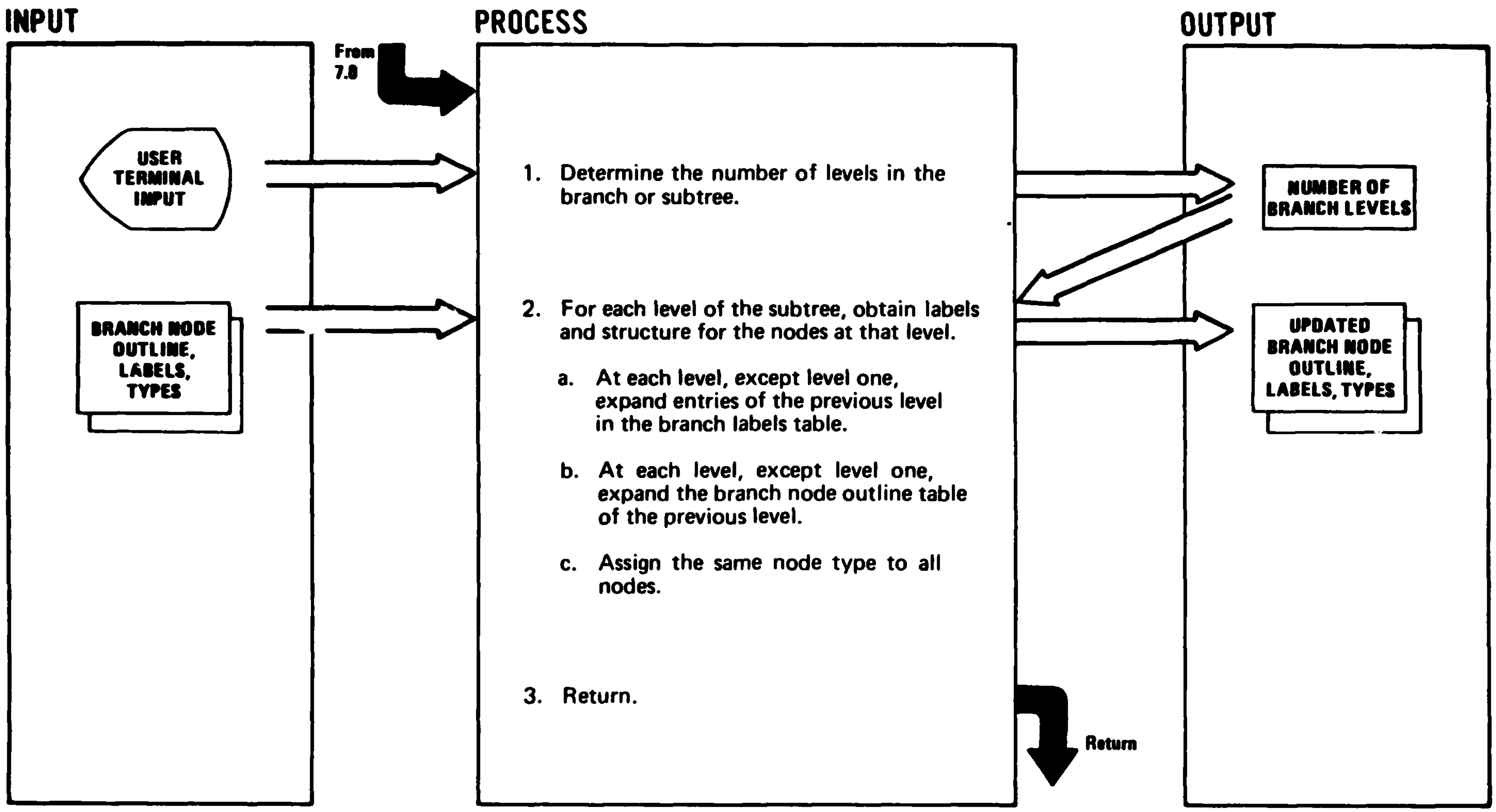
Diagram ID: 7.1 Description Create a Non-Symmetric Branch Page: of



1. Obtain branch node identifiers from the user. If no identifier is entered, do step 3.
2. Add entries to the branch node, label and type tables. Repeat from step 1.
3. Sort nodes in the branch or subtree into numerical sequence order.
4. Return.

System/Program: STRUCTURE Name:

Diagram ID: 7.2 Description Create a Symmetric Branch Page: of


INPUT
PROCESS
OUTPUT
From 7.0
USER TERMINAL INPUT
BRANCH NODE OUTLINE, LABELS, TYPES
NUMBER OF BRANCH LEVELS
UPDATED BRANCH NODE OUTLINE, LABELS, TYPES
Return

1. Determine the number of levels in the branch or subtree.

2. For each level of the subtree, obtain labels and structure for the nodes at that level.

   a. At each level, except level one, expand entries of the previous level in the branch labels table.

   b. At each level, except level one, expand the branch node outline table of the previous level.

   c. Assign the same node type to all nodes.

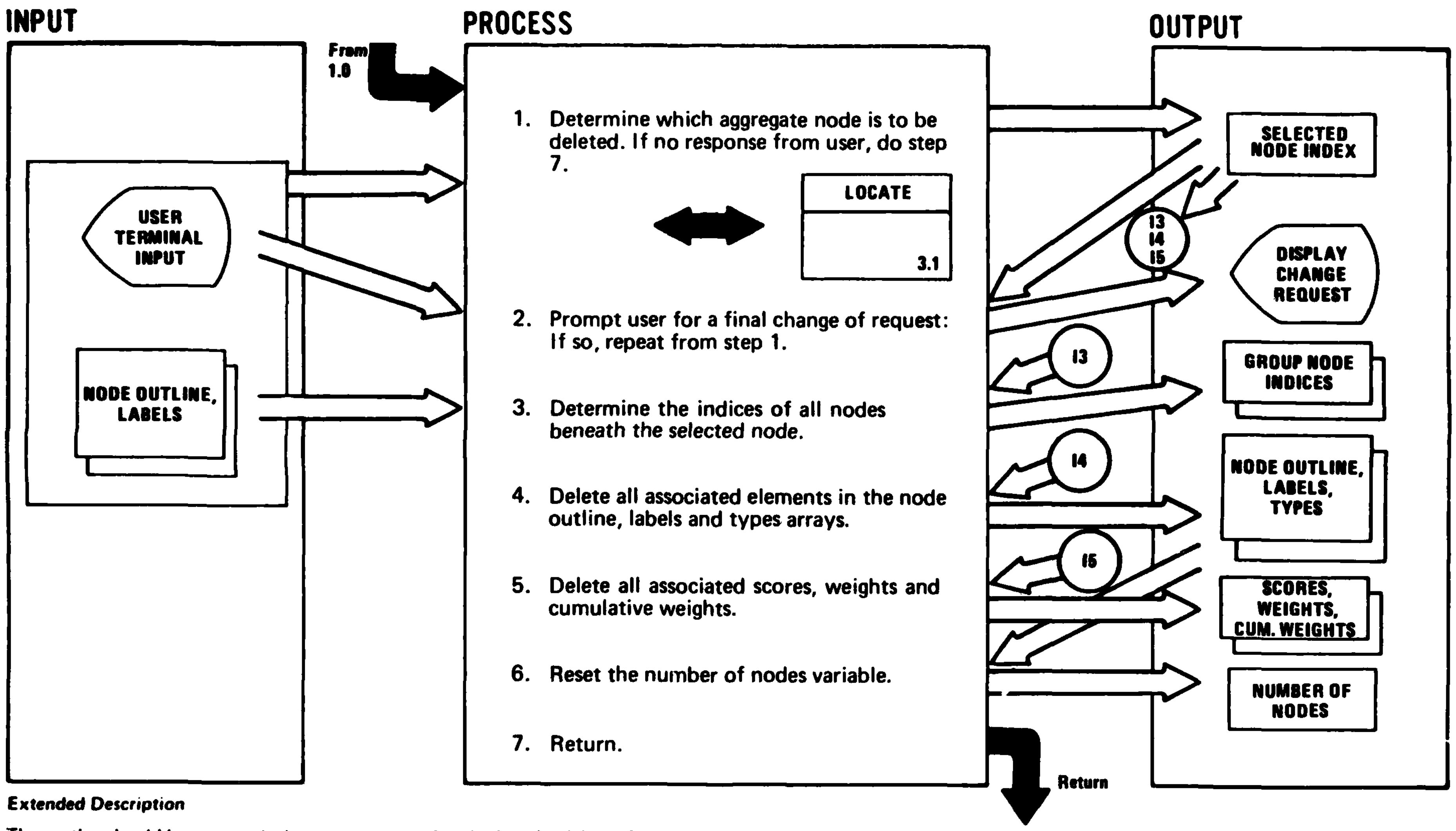
3. Return.

**Extended Description**

Step 2 processing ensures that for each subsequent level of a multilevel branch structure the outline number, types and labels are all added in the correct numerical sequence to the outline, types and label entries at the previous level. (This is done for every branch node defined at the previous level.)

System/Program: STRUCTURE Name:

Diagram ID: 8.0 Description Prune a Structure Section Page: of

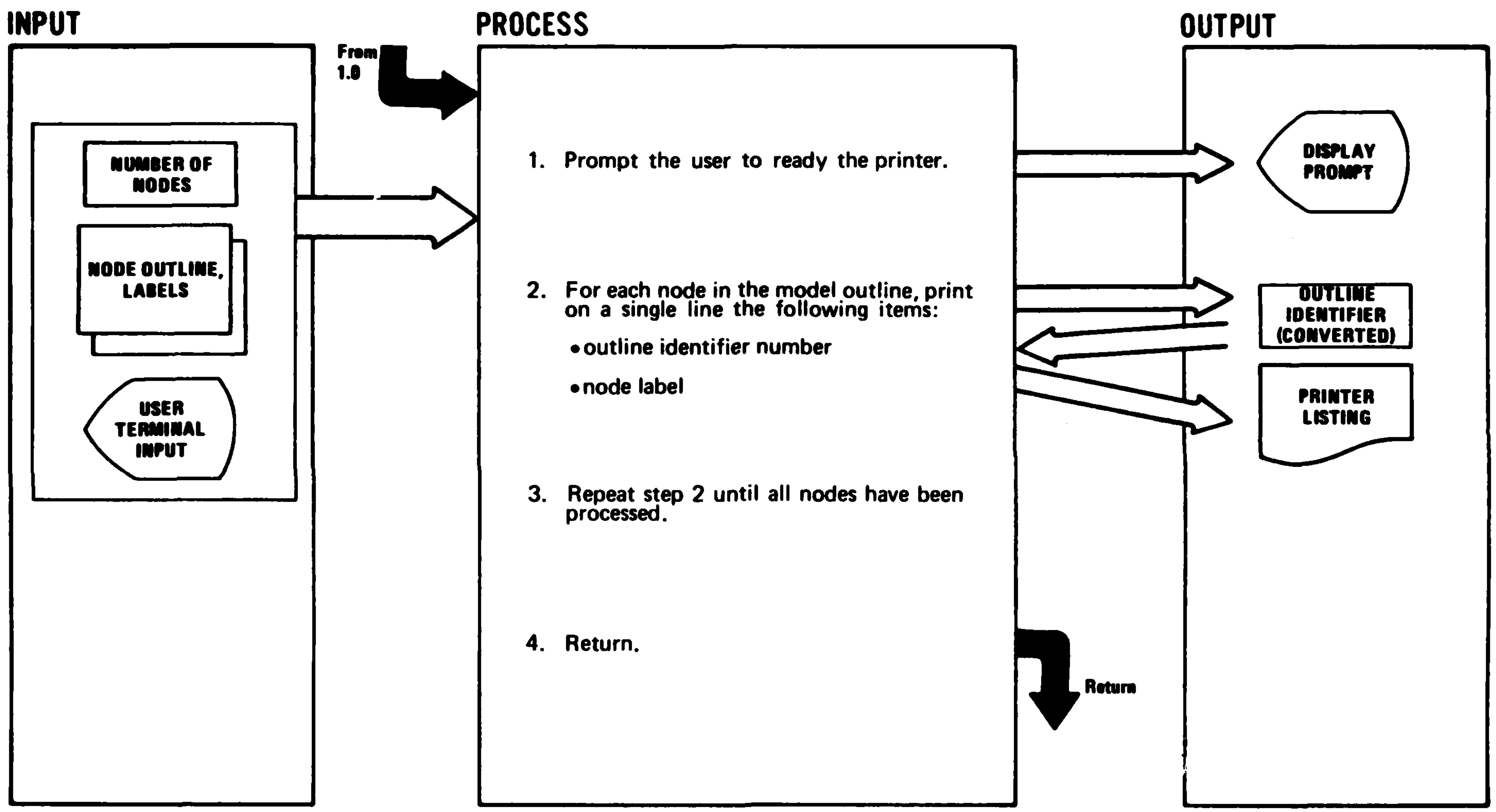


1. Determine which aggregate node is to be deleted. If no response from user, do step 7.
2. Prompt user for a final change of request: If so, repeat from step 1.
3. Determine the indices of all nodes beneath the selected node.
4. Delete all associated elements in the node outline, labels and types arrays.
5. Delete all associated scores, weights and cumulative weights.
6. Reset the number of nodes variable.
7. Return.

*Extended Description*

The routine should be executed whenever a group of nodes is to be deleted from an existing node structure. The grouped nodes are all hierarchically placed below a certain aggregate node; hence, a user specification of an aggregate node in step 1 will cause that node and all its subsequent nodes to be deleted.

System/Program: STRUCTURE Name: 
Diagram ID: 9.0 Description: Print a Review Sheet Page: ___ of ___



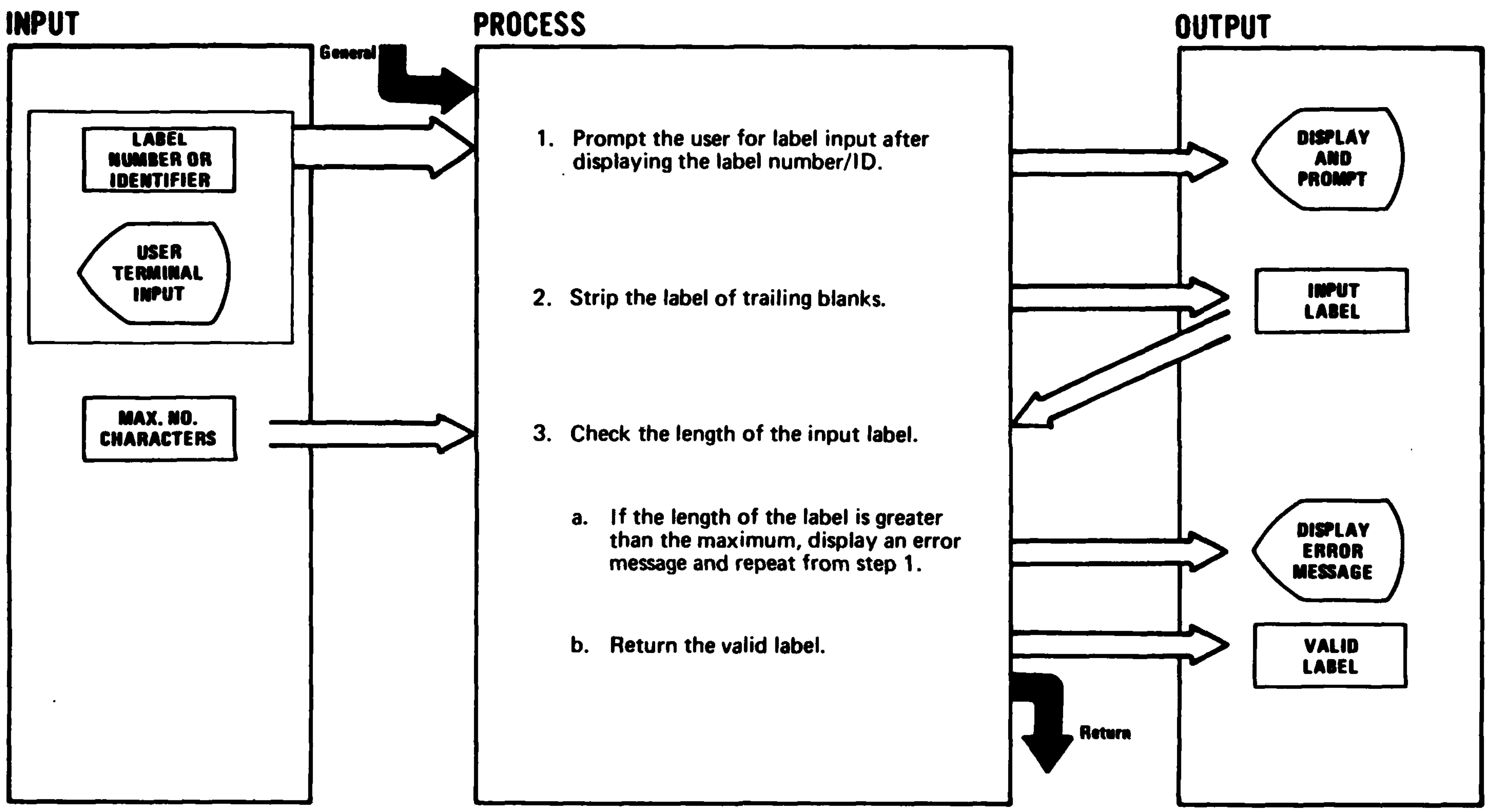
1. Prompt the user to ready the printer.
2. For each node in the model outline, print on a single line the following items:
   - outline identifier number
   - node label
3. Repeat step 2 until all nodes have been processed.
4. Return.

**Extended Description**

2. The decoded outline identifier number is formatted for output. The output should be equivalent to the user's original input during the creation of the structure.

System/Program: STRUCTURE Name: ENTER LABELS

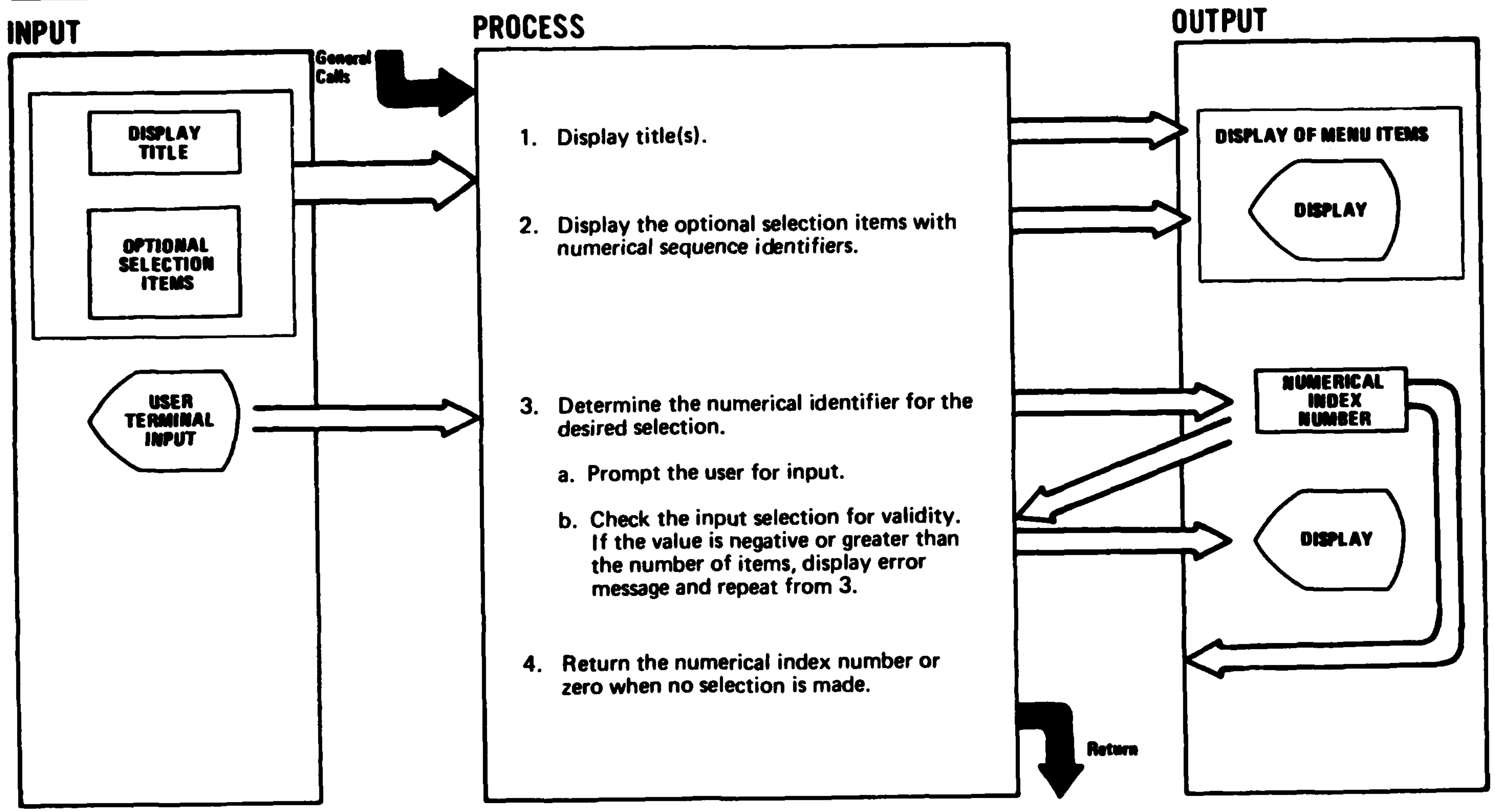
Diagram ID: 10.1 Description Enter Labels Routine Page: ___ of ___



1. Prompt the user for label input after displaying the label number/ID.
2. Strip the label of trailing blanks.
3. Check the length of the input label.
   a. If the length of the label is greater than the maximum, display an error message and repeat from step 1.
   b. Return the valid label.

System/Program: STRUCTURE Name: MENU

Diagram ID: 10.2 Description Display a Menu Page: —— of ——



1. Display title(s).
2. Display the optional selection items with numerical sequence identifiers.
3. Determine the numerical identifier for the desired selection.
   a. Prompt the user for input.
   b. Check the input selection for validity. If the value is negative or greater than the number of items, display error message and repeat from 3.
4. Return the numerical index number or zero when no selection is made.

**Extended Description**

1. The title is passed to this routine so that the display will remain in context with the processing function. For example, a title may be 'DISPLAY RESULTS.'

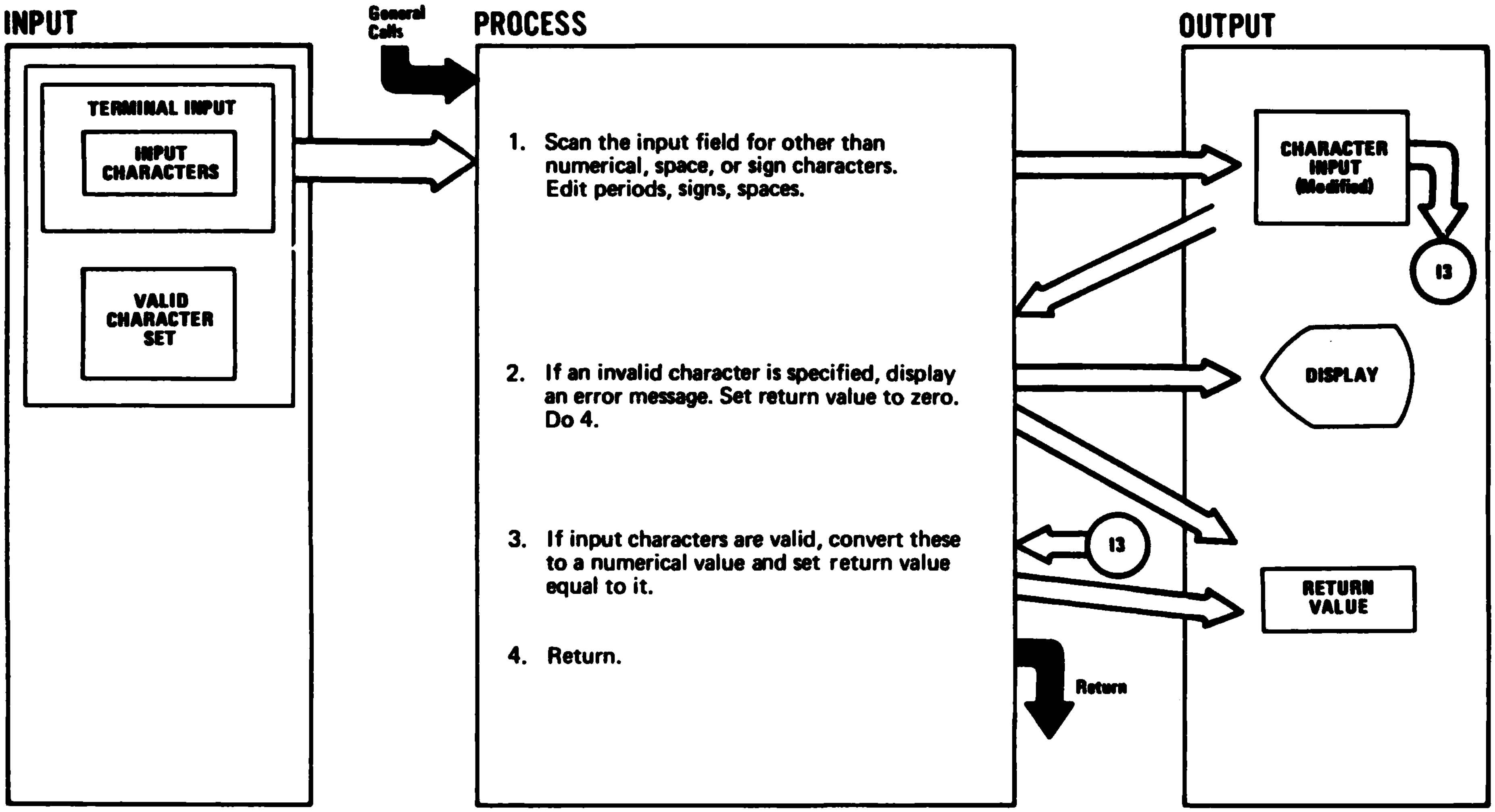
2. The selections that describe what is optimal are passed as input and are displayed in a list or cookbook MENU format along with item sequence numbers.

3. Prompt the user for the item sequence number of the choice selection.

Check the validity of the user input.

System/Program: STRUCTURE Name:

Diagram ID: 10.3 Description Select Only Numeric Input Page: of



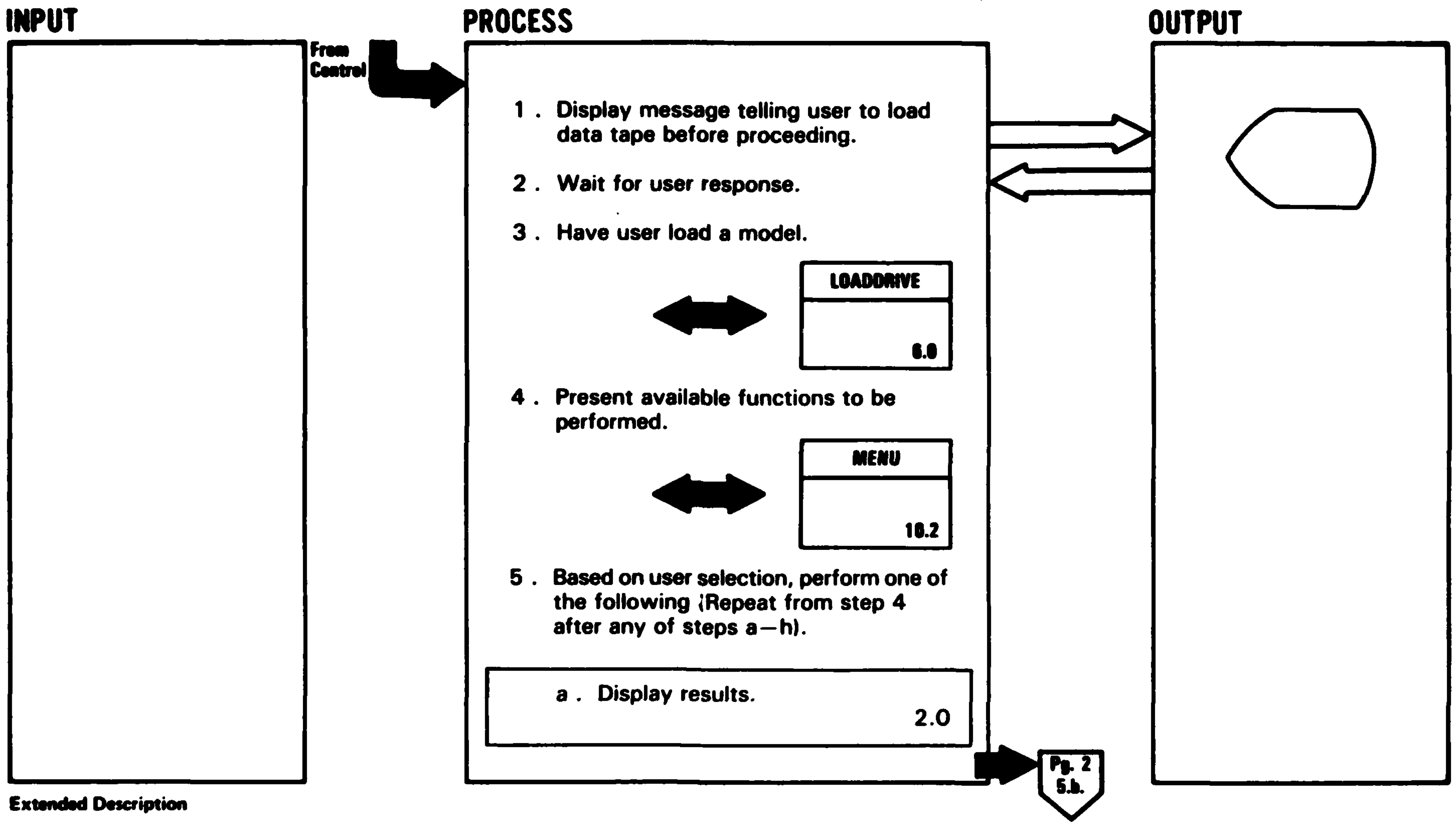
1. Scan the input field for other than numerical, space, or sign characters. Edit periods, signs, spaces.
2. If an invalid character is specified, display an error message. Set return value to zero. Do 4.
3. If input characters are valid, convert these to a numerical value and set return value equal to it.
4. Return.

**Extended Description**

This routine will not be required if system error checking routines interface with the standard keyboard-display input.

System/Program: RUN Name: RUN

Diagram ID: 1.0 Description Entrance to Eval Program Page: 1 of 2



1 . Display message telling user to load data tape before proceeding.

2 . Wait for user response.

3 . Have user load a model.

4 . Present available functions to be performed.

5 . Based on user selection, perform one of the following (Repeat from step 4 after any of steps a—h).

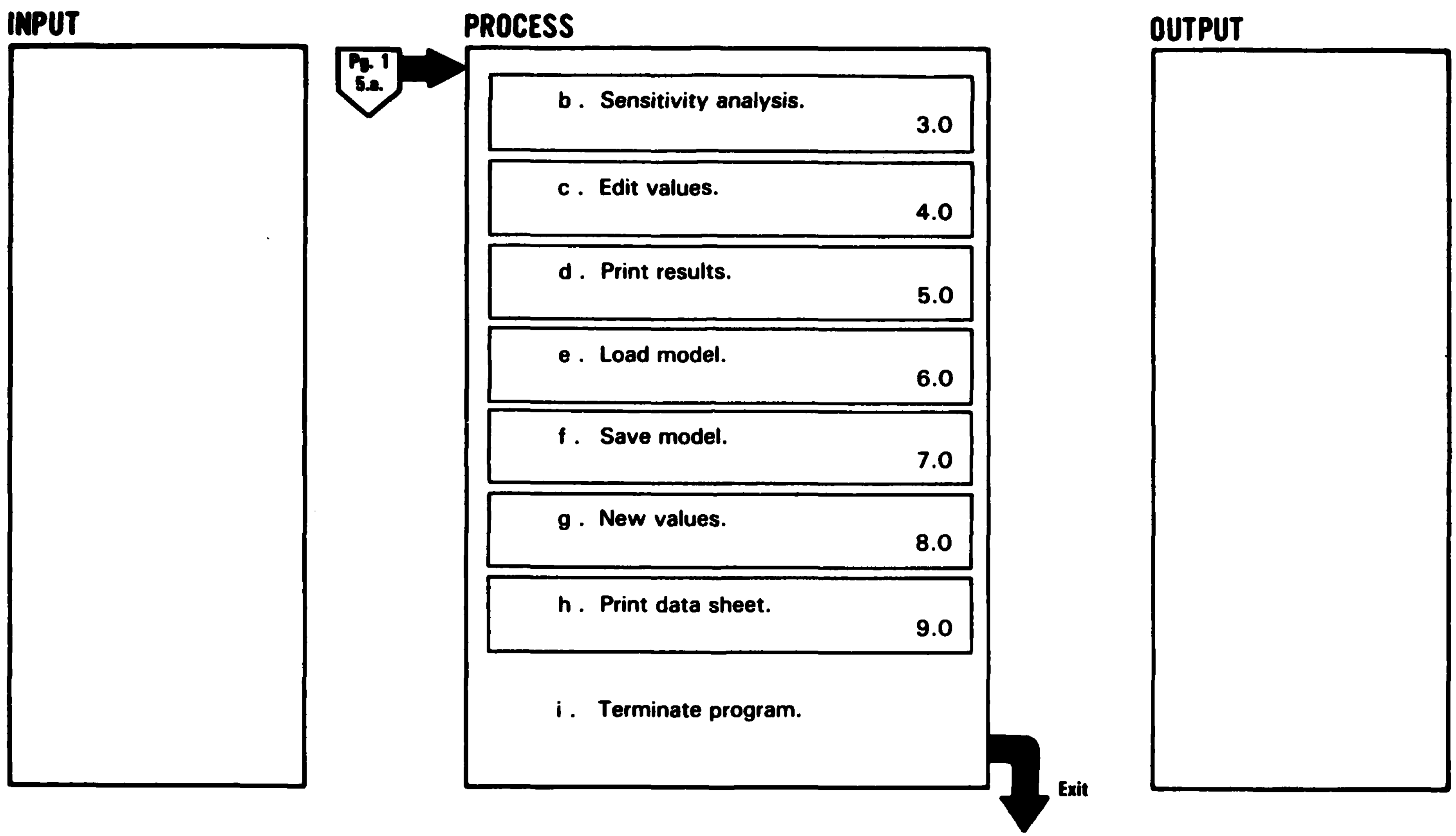
a . Display results. 2.0

**Extended Description**

3. The model variables are all loaded into the current work area at this time, or whenever the user wishes to load a new model. Consequently, this documentation assumes that these variables are "global" and always available for reference, input to procedures, or modification.

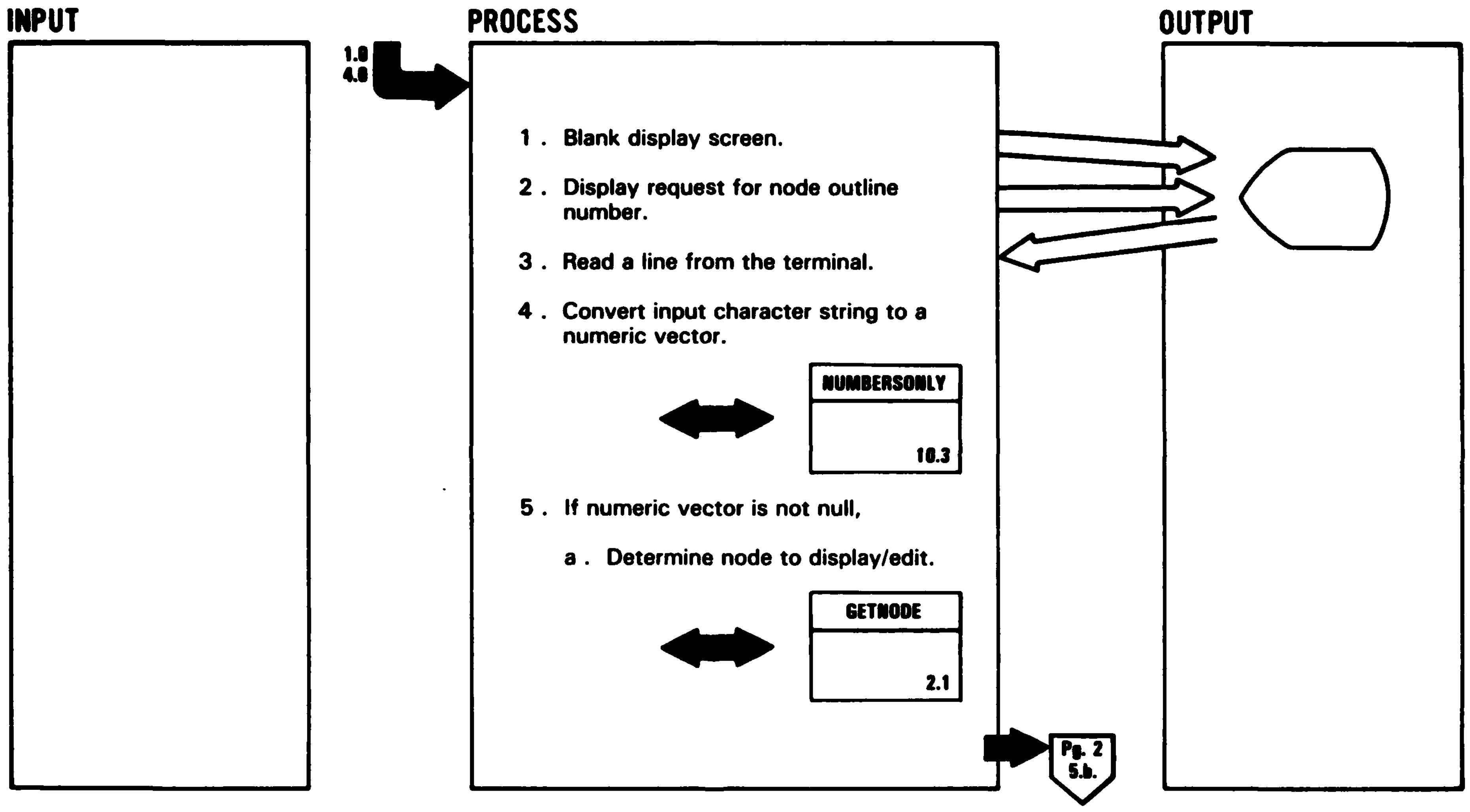
System/Program: RUN Name: RUN

Diagram ID: 1.0 Description Entrance to Eval Program Page: 2 of 2



System/Program: RUN Name: SELECT

Diagram ID: 2.0 Description Elicit & Display/Edit Node Page: 1 of 2



1 . Blank display screen.

2 . Display request for node outline number.

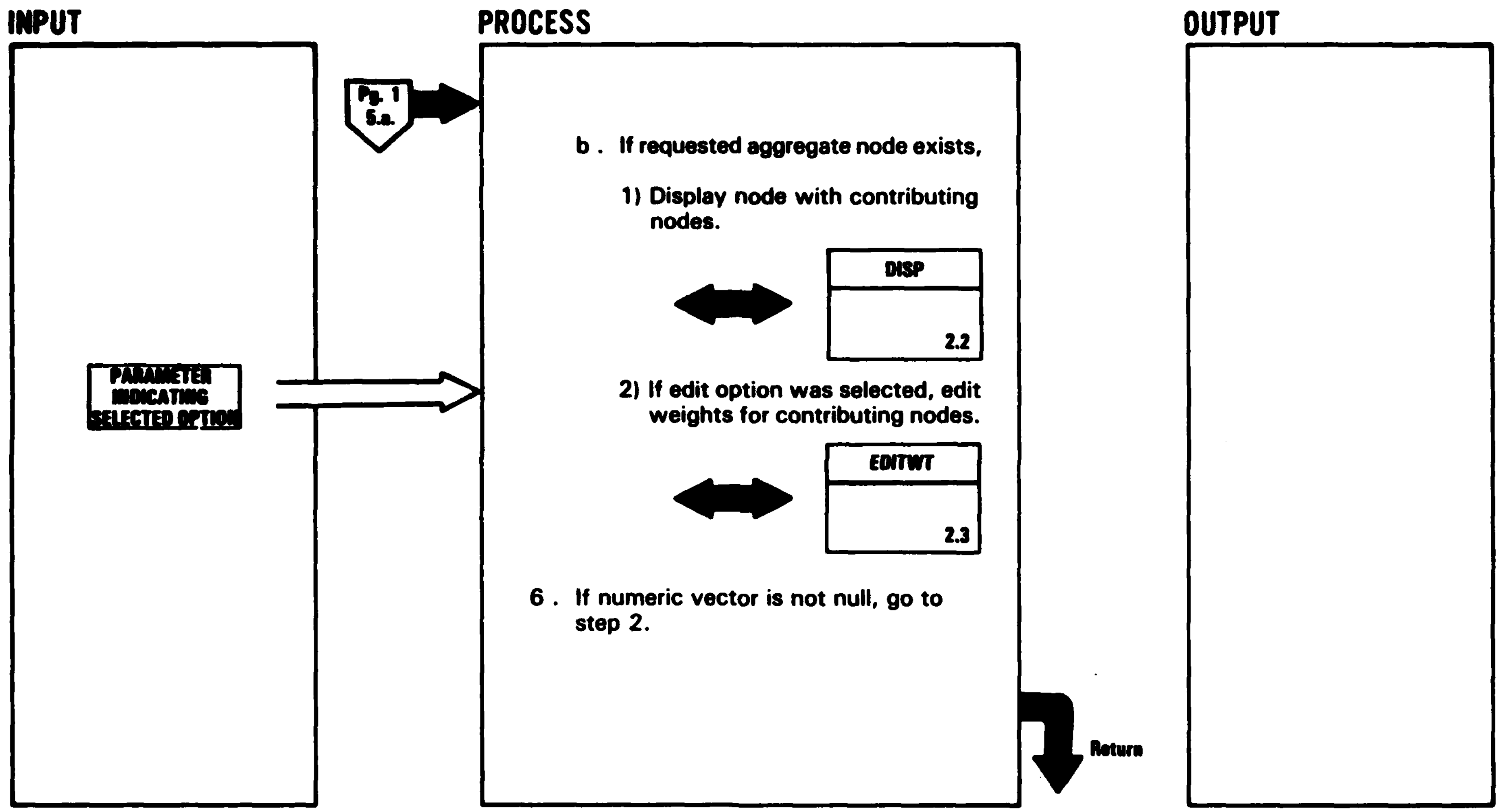
3 . Read a line from the terminal.

4 . Convert input character string to a numeric vector.

5 . If numeric vector is not null,

a . Determine node to display/edit.

System/Program: RUN Name: SELECT
Diagram ID: 2.0 Description Elicit & Display/Edit Node Page: 2 of 2



b . If requested aggregate node exists,

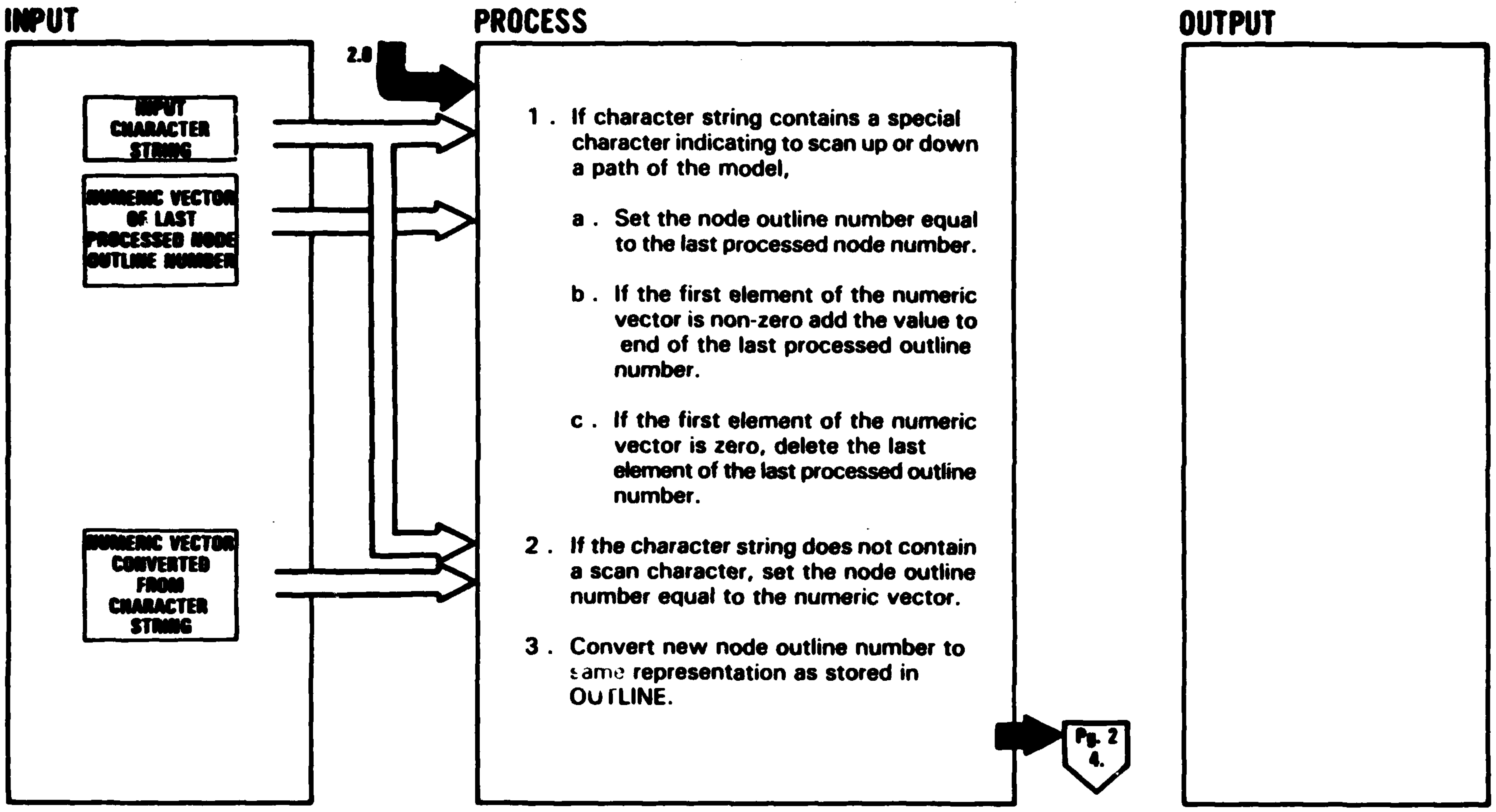
1) Display node with contributing nodes.

2) If edit option was selected, edit weights for contributing nodes.

6 . If numeric vector is not null, go to step 2.

System/Program: RUN Name: GETNODE

Diagram ID: 2.1 Description Determine Aggregate Node to Display Page: 1 of 2


INPUT
PROCESS
OUTPUT
INPUT CHARACTER STRING
NUMERIC VECTOR OF LAST PROCESSED NODE OUTLINE NUMBER
NUMERIC VECTOR CONVERTED FROM CHARACTER STRING
2.0
Pg. 2 4.

1 . If character string contains a special character indicating to scan up or down a path of the model,

   a . Set the node outline number equal to the last processed node number.

   b . If the first element of the numeric vector is non-zero add the value to end of the last processed outline number.

   c . If the first element of the numeric vector is zero, delete the last element of the last processed outline number.

2 . If the character string does not contain a scan character, set the node outline number equal to the numeric vector.

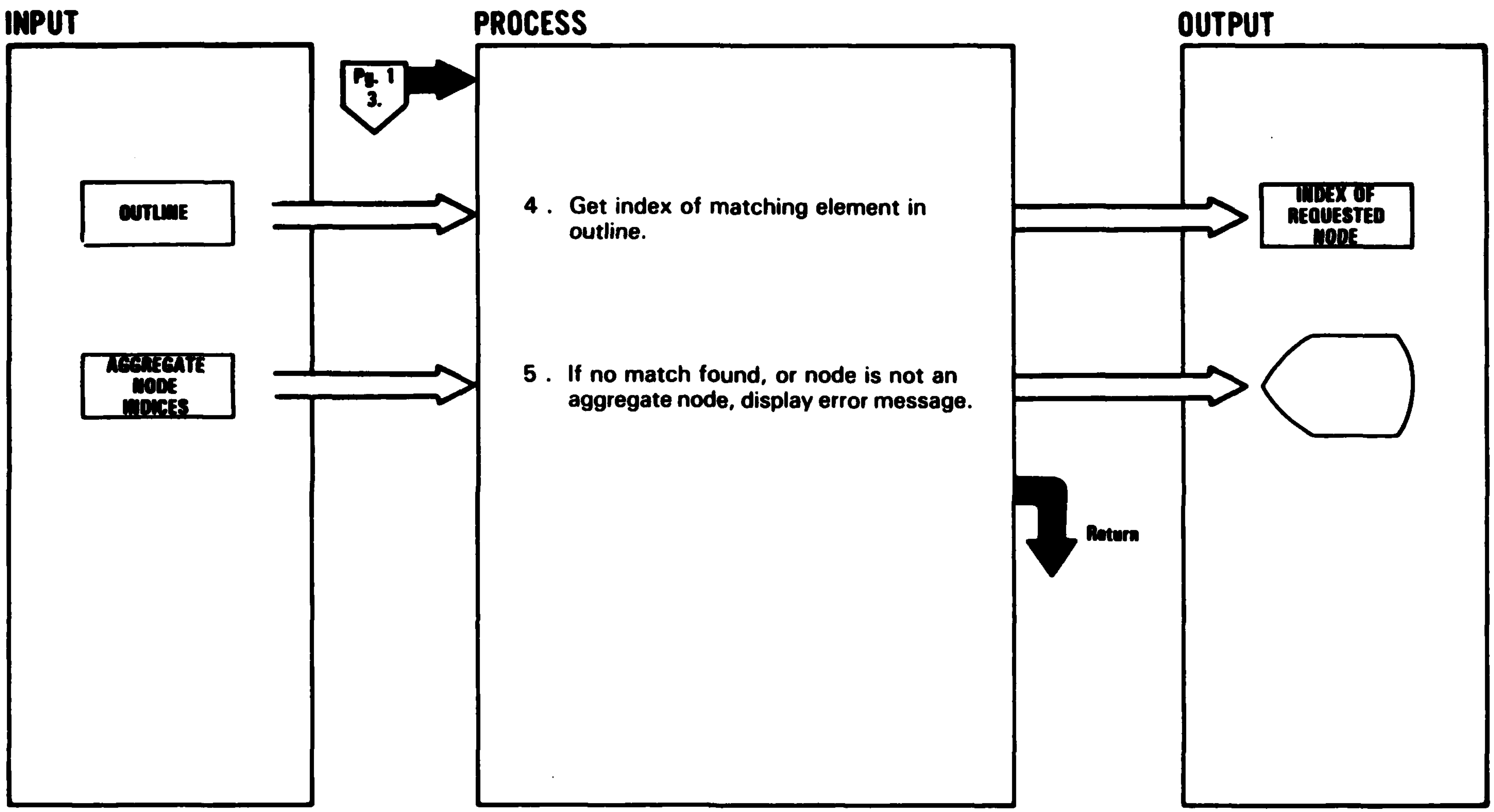
3 . Convert new node outline number to same representation as stored in OUTLINE.

**Extended Description**

b. This generates a node outline number one level deeper than the previously processed node. For example, if the previously processed number were 3.2.5 and the input '6)' (where the right parenthesis is the scan operator) the new node outline number would be 3.2.5.6.

c. This generates a node outline number one level higher than the previously processed node. For example, if the previously processed number were 3.2.5 and the input '0)' (where the right parenthesis is the scan operator), the new node outline number would be 3.2.

System/Program: RUN Name: GETNODE
Diagram ID: 2.1 Description Determine Aggregate Node to Display Page: 2 of 2

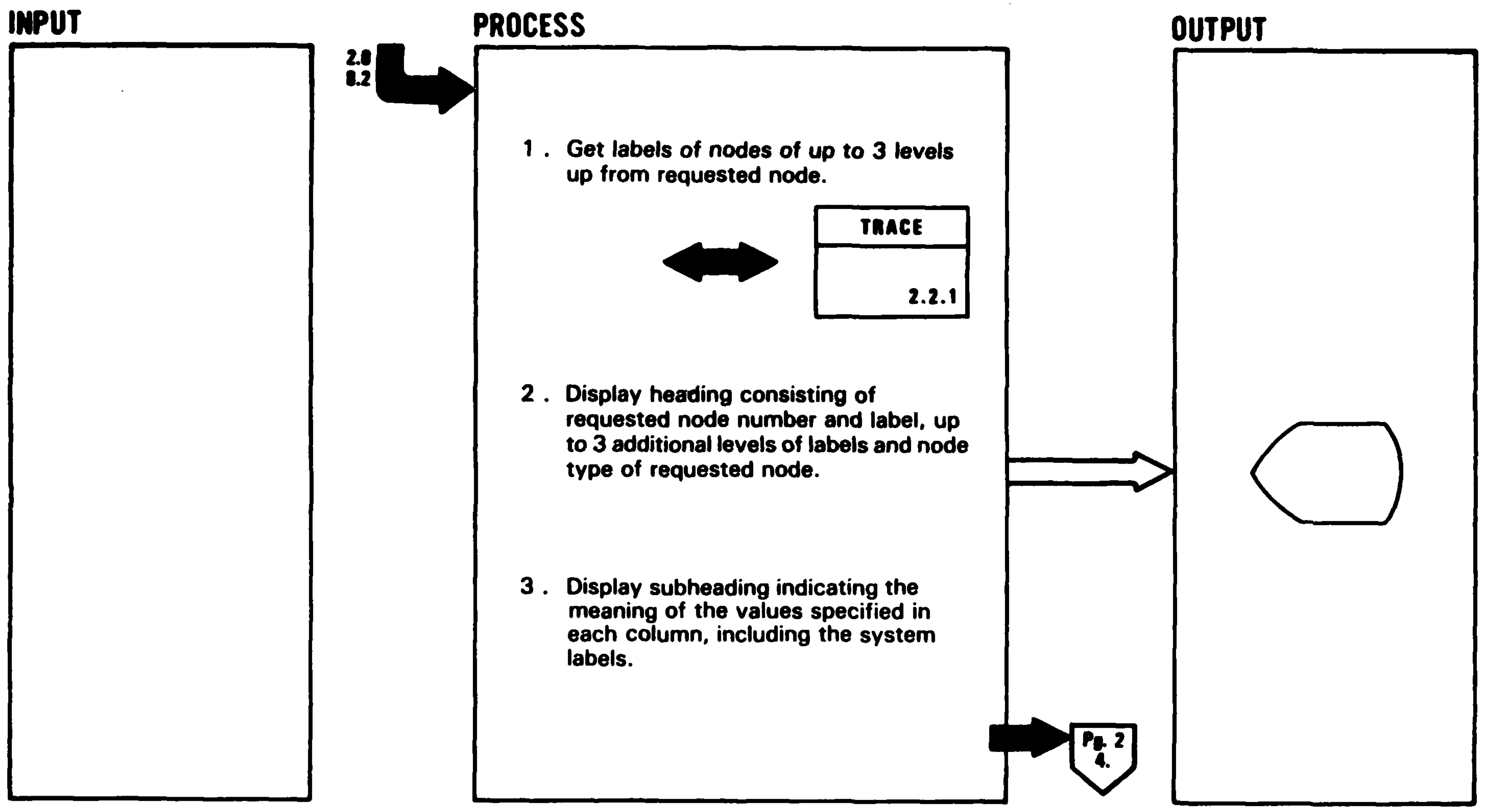


4 . Get index of matching element in outline.

5 . If no match found, or node is not an aggregate node, display error message.

System/Program: RUN Name: DISP

Diagram ID: 2.2 Description Display Group Page: 1 of 2



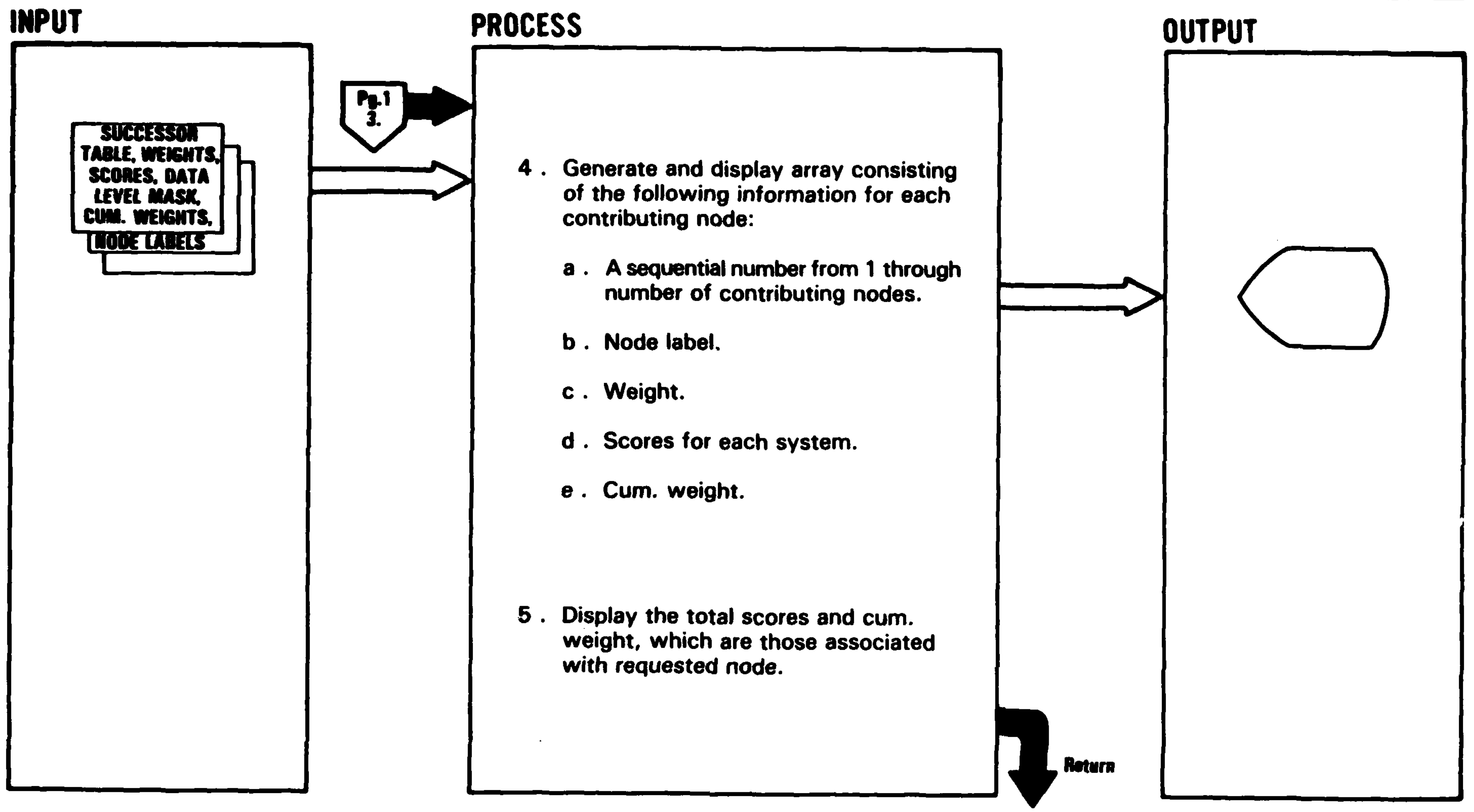
1 . Get labels of nodes of up to 3 levels up from requested node.

2 . Display heading consisting of requested node number and label, up to 3 additional levels of labels and node type of requested node.

3 . Display subheading indicating the meaning of the values specified in each column, including the system labels.

System/Program: RUN Name: DISP

Diagram ID: 2.2 Description Display Group Page: 2 of 2


INPUT
PROCESS
OUTPUT
SUCCESSOR TABLE, WEIGHTS, SCORES, DATA LEVEL MASK, CUM. WEIGHTS, NODE LABELS
Pg.1 3.
Return

4 . Generate and display array consisting of the following information for each contributing node:

a . A sequential number from 1 through number of contributing nodes.

b . Node label.

c . Weight.

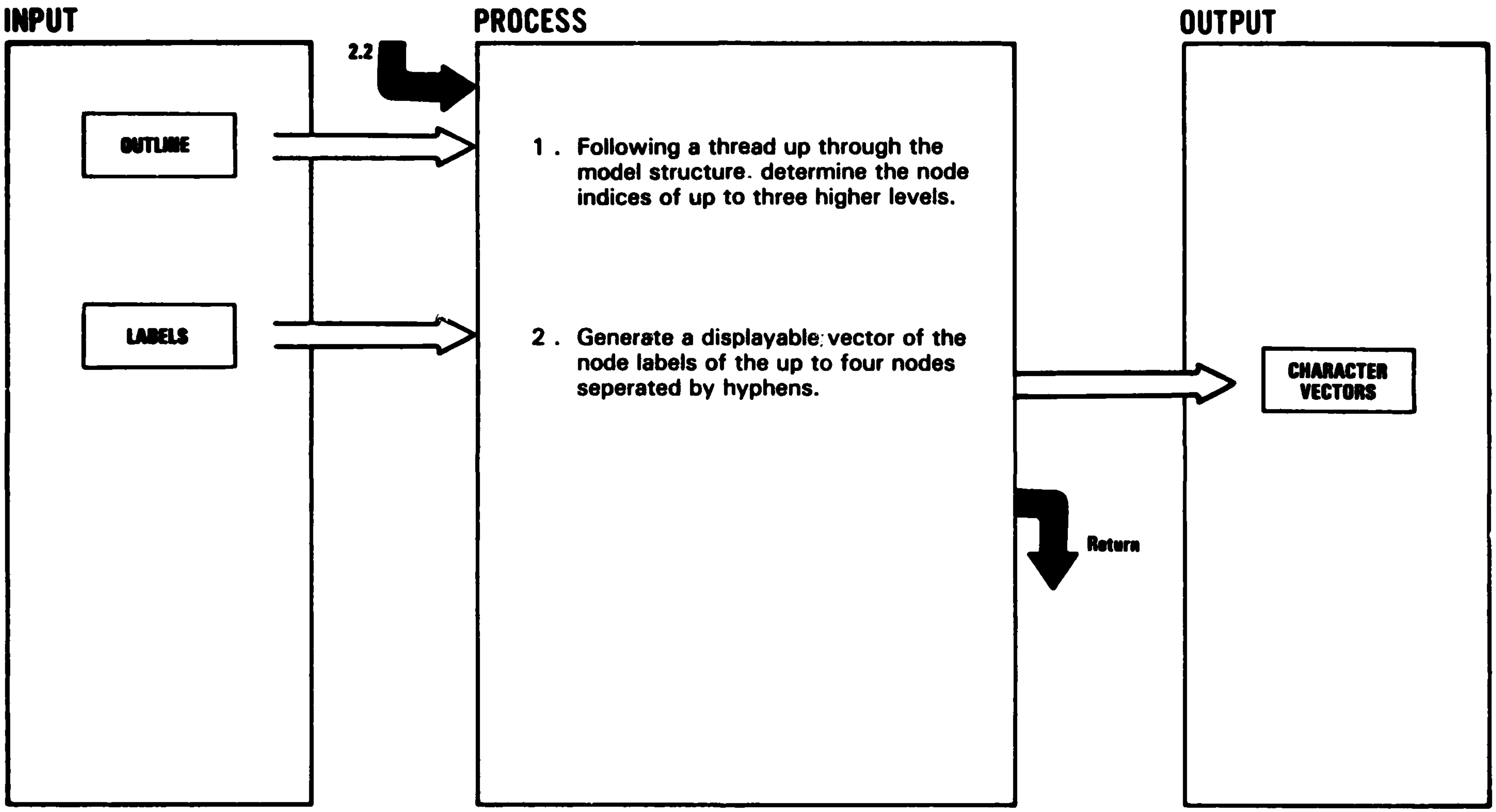
d . Scores for each system.

e . Cum. weight.

5 . Display the total scores and cum. weight, which are those associated with requested node.

System/Program: RUN Name: TRACE

Diagram ID: 2.2.1 Description Get Nodes of Thread Up Tree Page: ___ of ___



1 . Following a thread up through the model structure. determine the node indices of up to three higher levels.

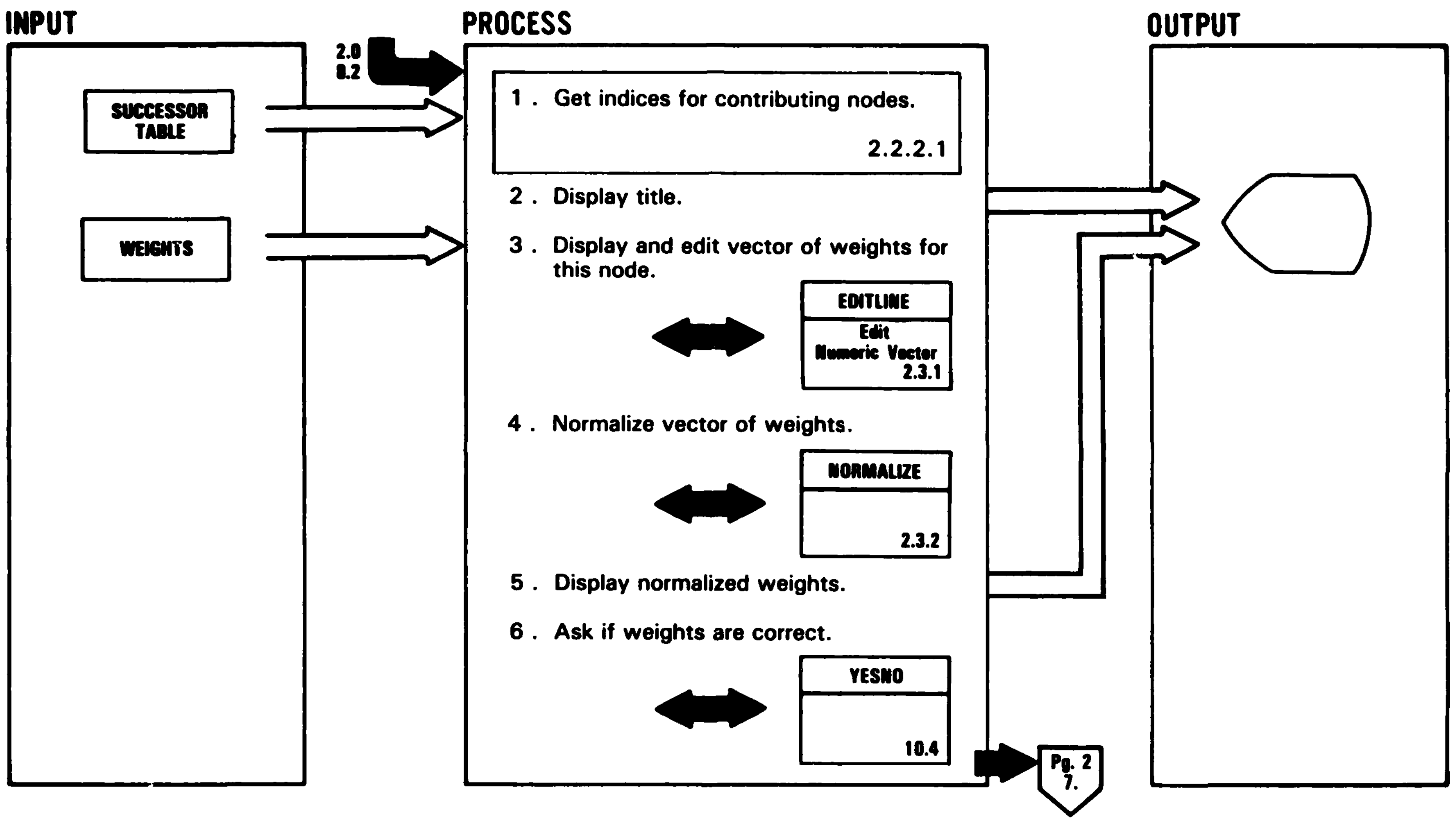
2 . Generate a displayable vector of the node labels of the up to four nodes seperated by hyphens.

**Extended Description**

For instance, if the requested node number is 1.4.2.6, the next higher level would be 1.4.2, and the fourth (or highest) calculated level would be 1.4.

System/Program: RUN Name: EDITWT

Diagram ID: 2.3 Description Edit Weights Page: 1 of 2



1 . Get indices for contributing nodes. 2.2.2.1

2 . Display title.

3 . Display and edit vector of weights for this node.

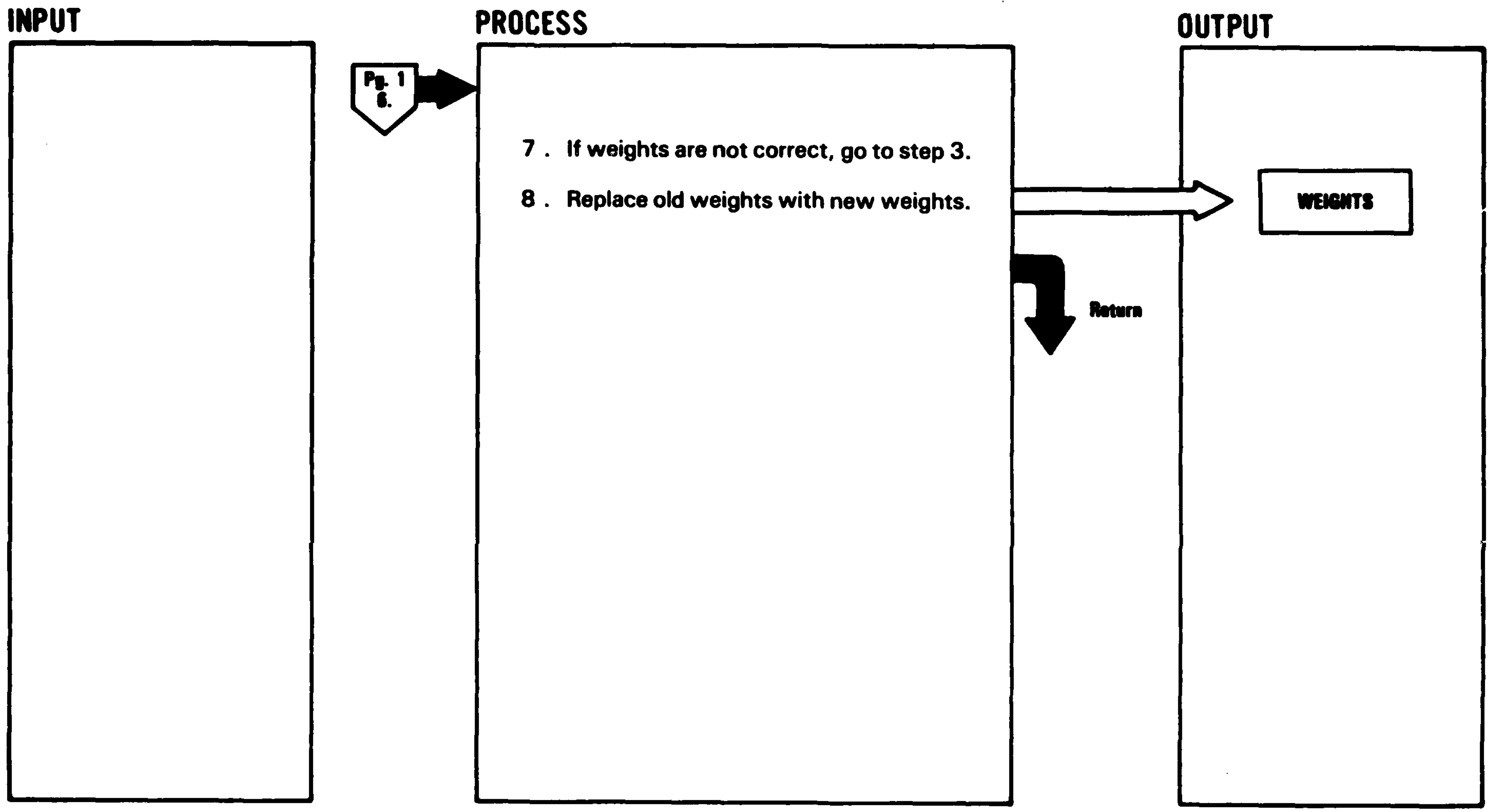
4 . Normalize vector of weights.

5 . Display normalized weights.

6 . Ask if weights are correct.

System/Program: RUN Name: EDITWT

Diagram ID: 2.3 Description Edit Weights Page: 2 of 2

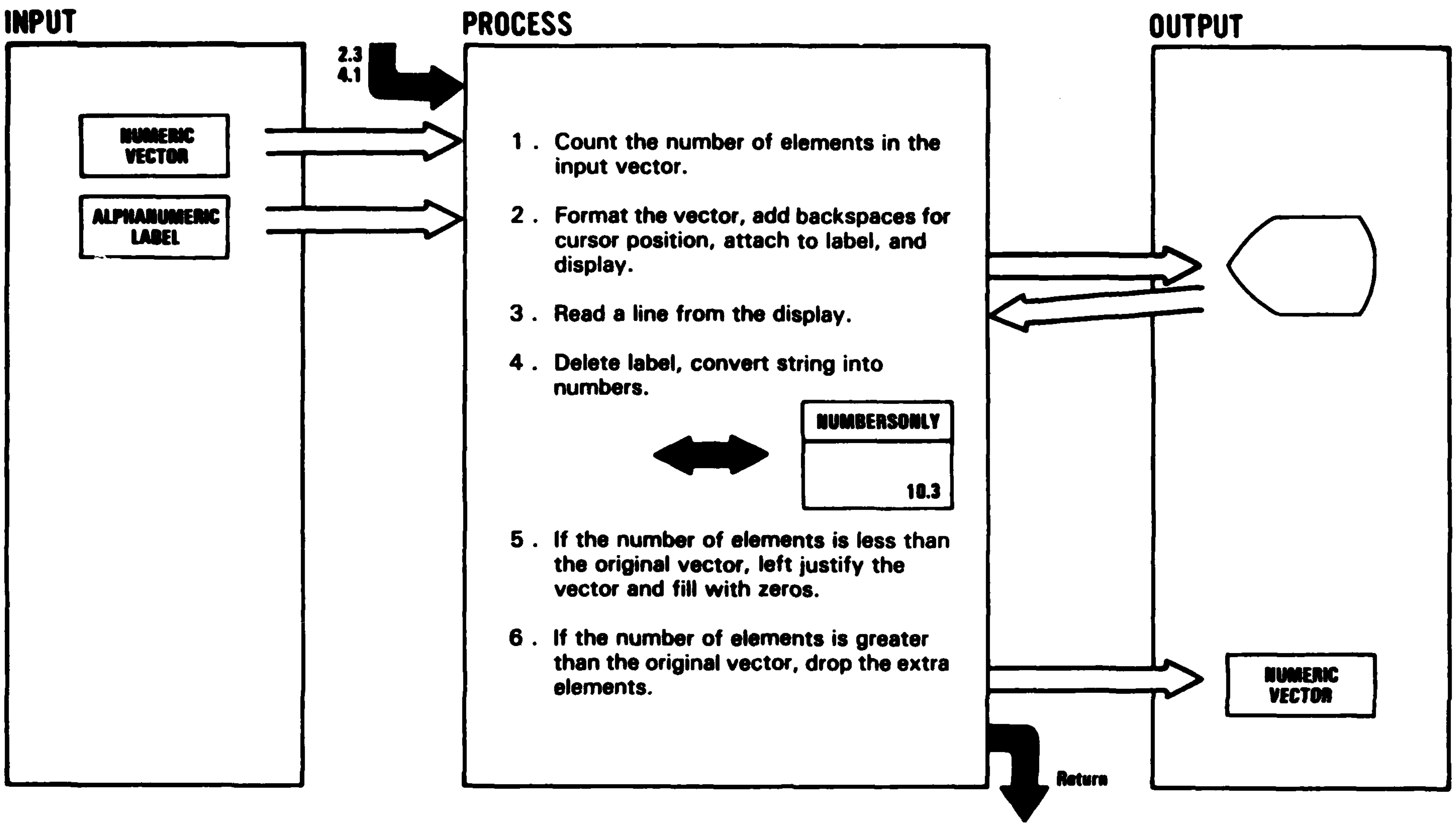


7 . If weights are not correct, go to step 3.

8 . Replace old weights with new weights.

System/Program: RUN Name: EDITLINE

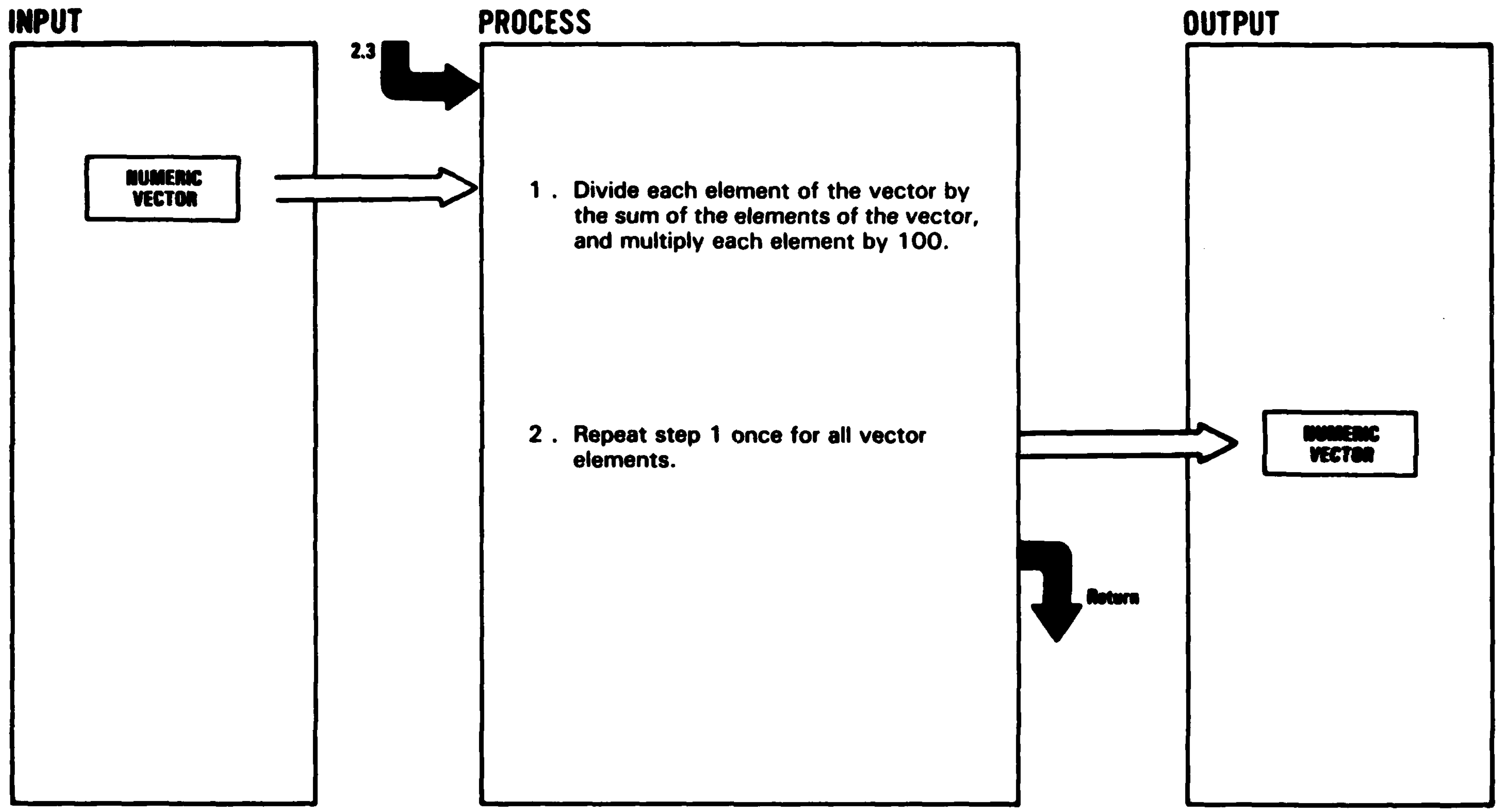
Diagram ID: 2.3.1 Description Edit a Numeric Vector Page: ____ of ____



1. Count the number of elements in the input vector.
2. Format the vector, add backspaces for cursor position, attach to label, and display.
3. Read a line from the display.
4. Delete label, convert string into numbers.
5. If the number of elements is less than the original vector, left justify the vector and fill with zeros.
6. If the number of elements is greater than the original vector, drop the extra elements.

System/Program: RUN Name: NORMALIZE

Diagram ID: 2.3.2 Description Normalize a Vector of Numbers Page: ___ of ___



1 . Divide each element of the vector by the sum of the elements of the vector, and multiply each element by 100.

2 . Repeat step 1 once for all vector elements.

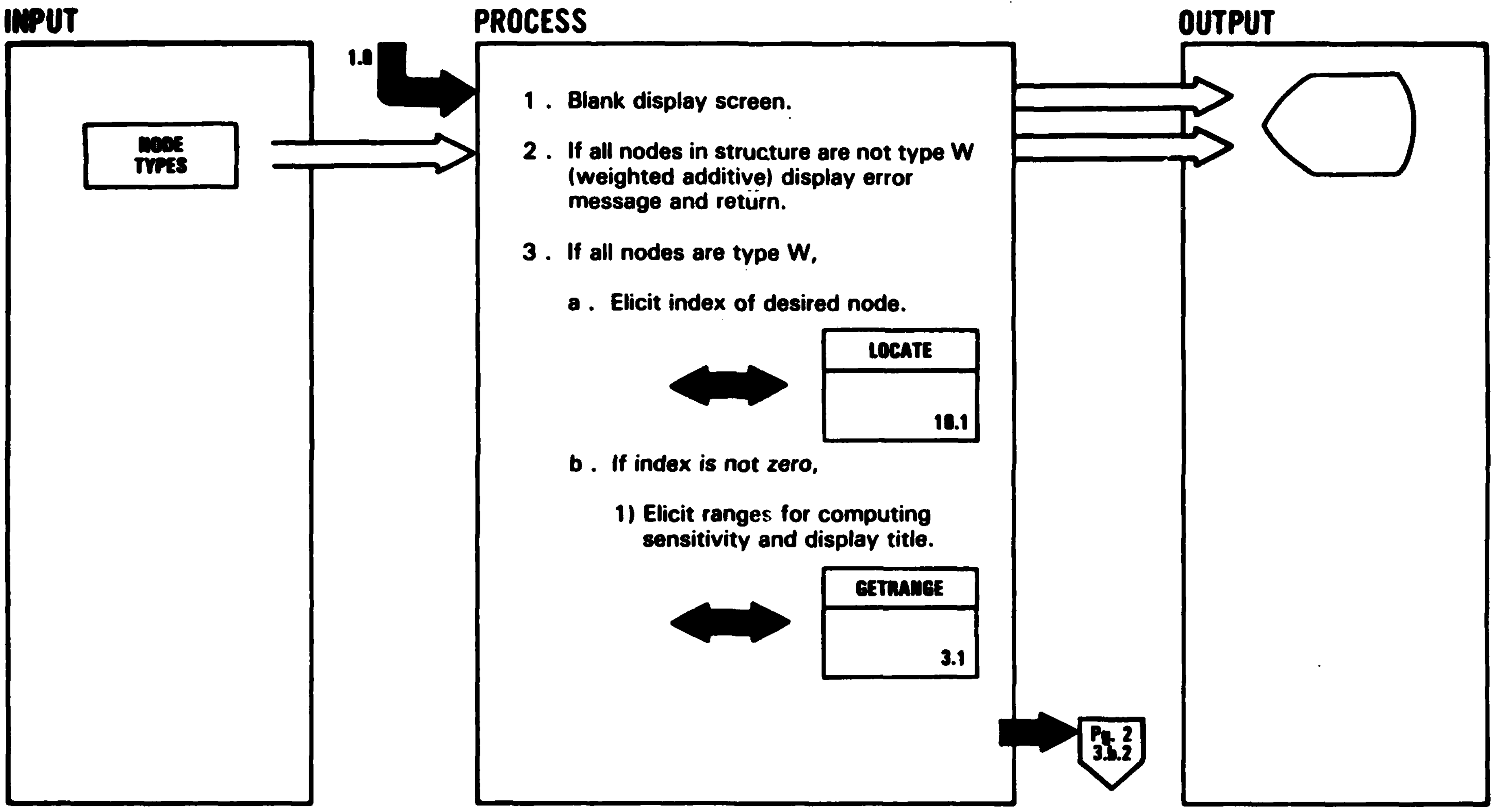
**Extended Description**

1. Performing this operation converts a group of arbitrary values to a group of values that add up to 100. The values all maintain the same relativity.

2. Performing this operation twice allows the case where the original values are all zero. The final result is a group of equal numbers that add up to 100.

System/Program: RUN  Name: SENSITIVITY

Diagram ID: 3.0  Description Perform Sensitivity Analysis  Page: 1 of 2



1 . Blank display screen.

2 . If all nodes in structure are not type W (weighted additive) display error message and return.

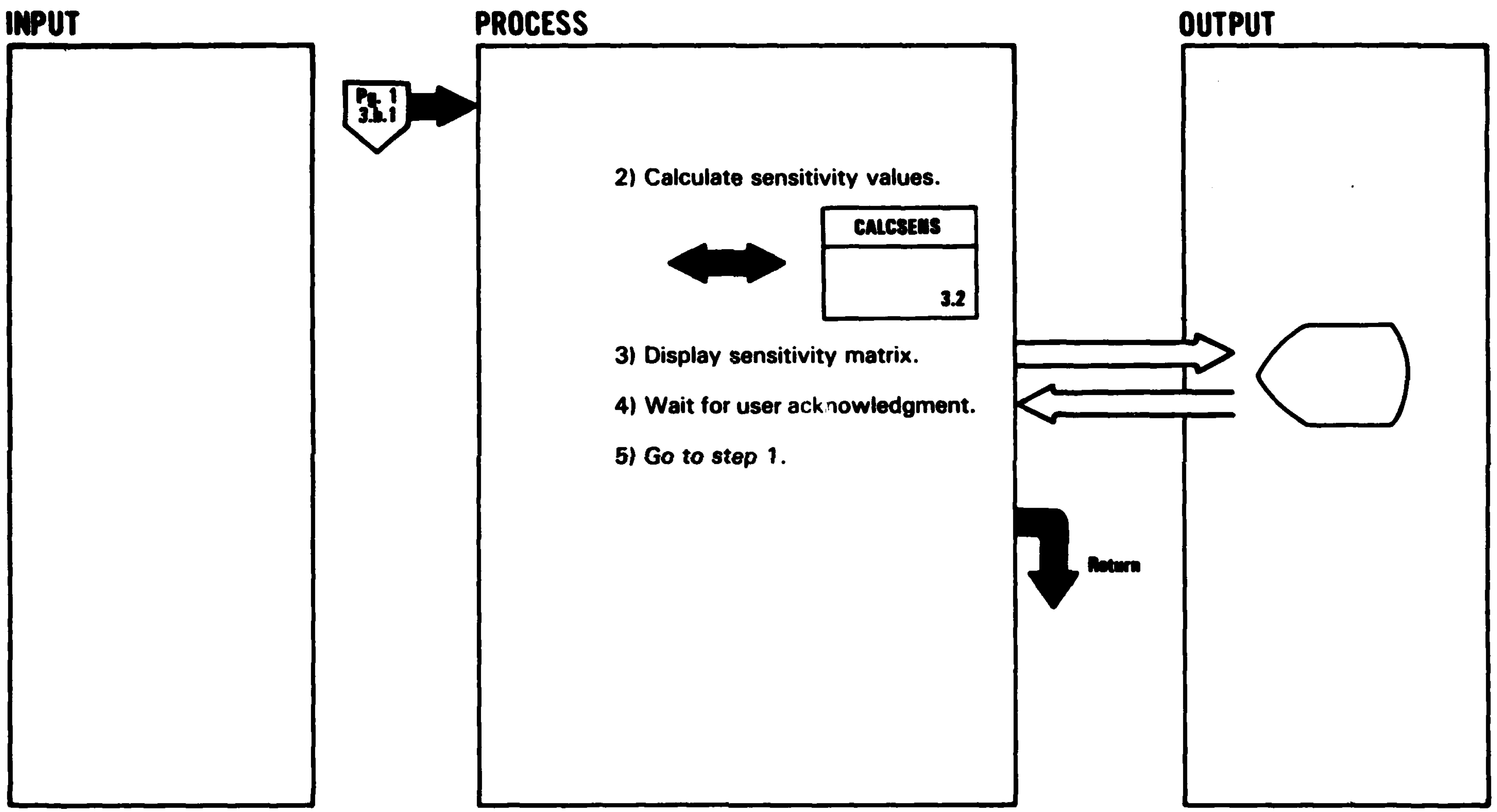
3 . If all nodes are type W,

a . Elicit index of desired node.

b . If index is not zero,

1) Elicit ranges for computing sensitivity and display title.

| | |
|---|---|
| System/Program: RUN | Name: SENSITIVITY |
| Diagram ID: 3.0 | Description Perform Sensitivity Analysis |
| Page: 2 of 2 | |



2) Calculate sensitivity values.

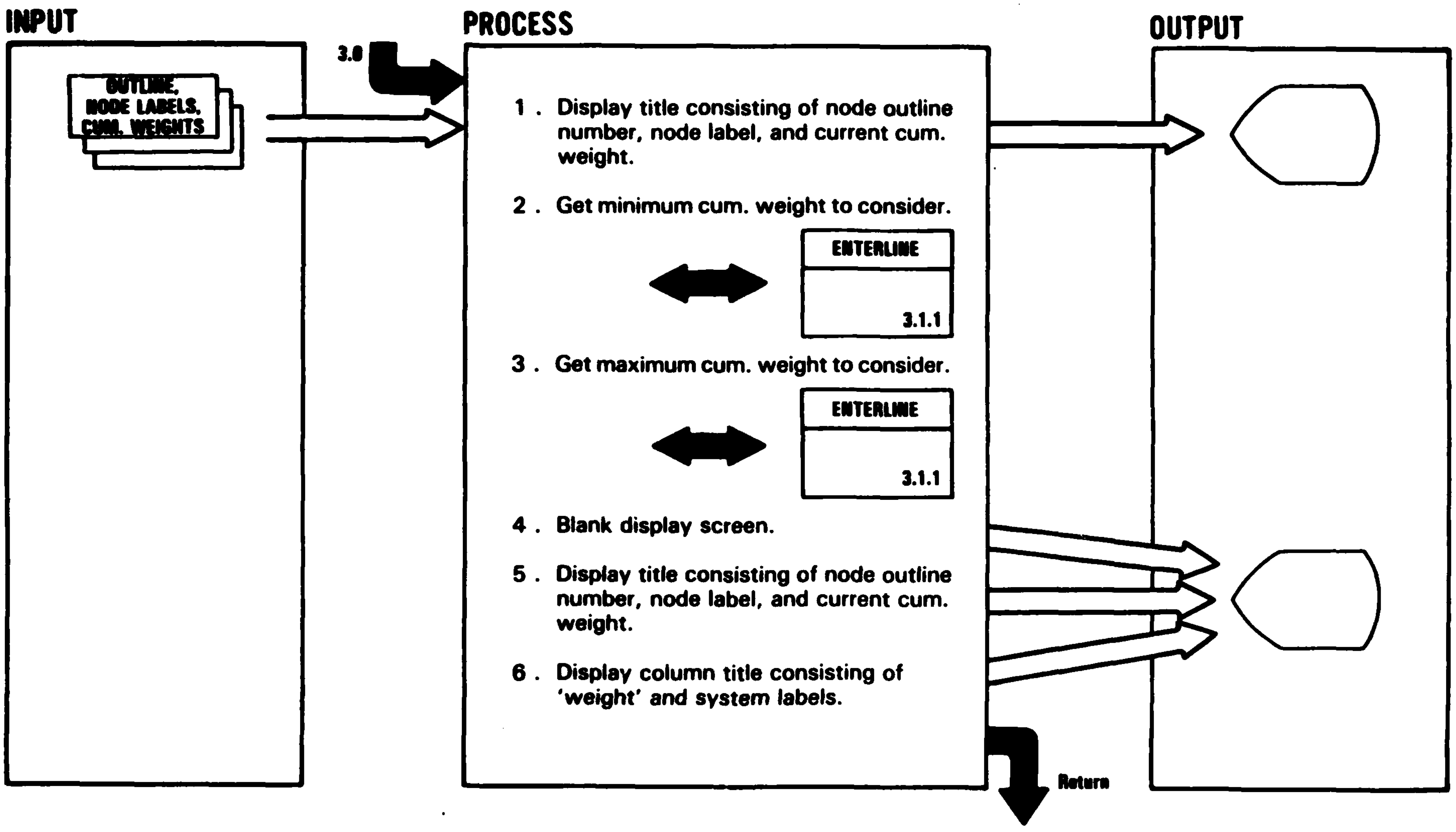
3) Display sensitivity matrix.

4) Wait for user acknowledgment.

5) *Go to step 1.*

System/Program: RUN Name: GETRANGE

Diagram ID: 3.1 Description Elicit Ranges for Computing Sensitivity Analysis Page: ___ of ___



1 . Display title consisting of node outline number, node label, and current cum. weight.

2 . Get minimum cum. weight to consider.

3 . Get maximum cum. weight to consider.

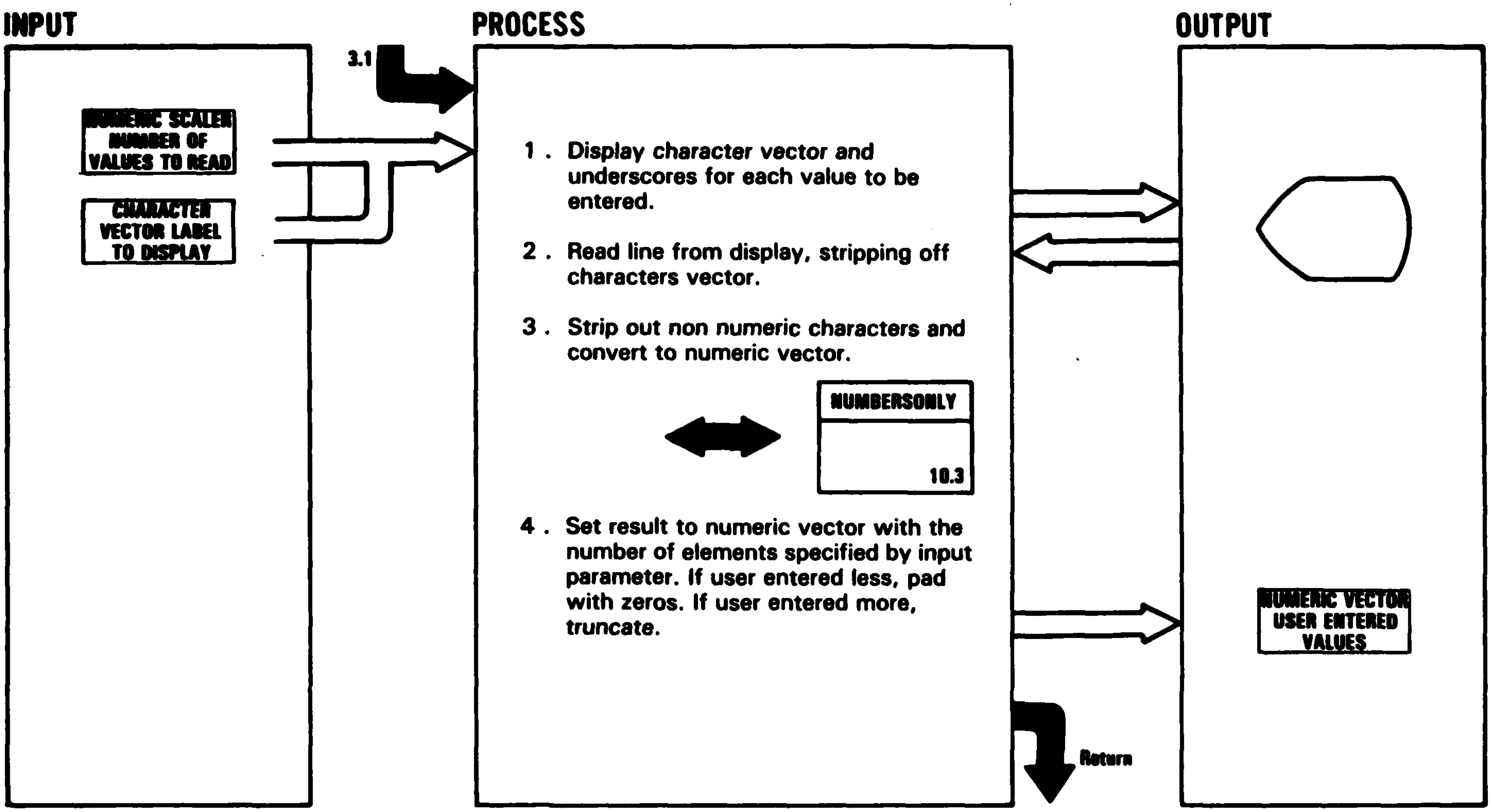
4 . Blank display screen.

5 . Display title consisting of node outline number, node label, and current cum. weight.

6 . Display column title consisting of 'weight' and system labels.

System/Program: RUN Name: ENTERLINE

Diagram ID: 3.1.1 Description Enter Numeric Vector Page: ____ of ____


INPUT
PROCESS
OUTPUT
NUMERIC SCALER NUMBER OF VALUES TO READ
CHARACTER VECTOR LABEL TO DISPLAY
3.1
NUMBERSONLY
10.3
NUMERIC VECTOR USER ENTERED VALUES
Return

1 . Display character vector and underscores for each value to be entered.

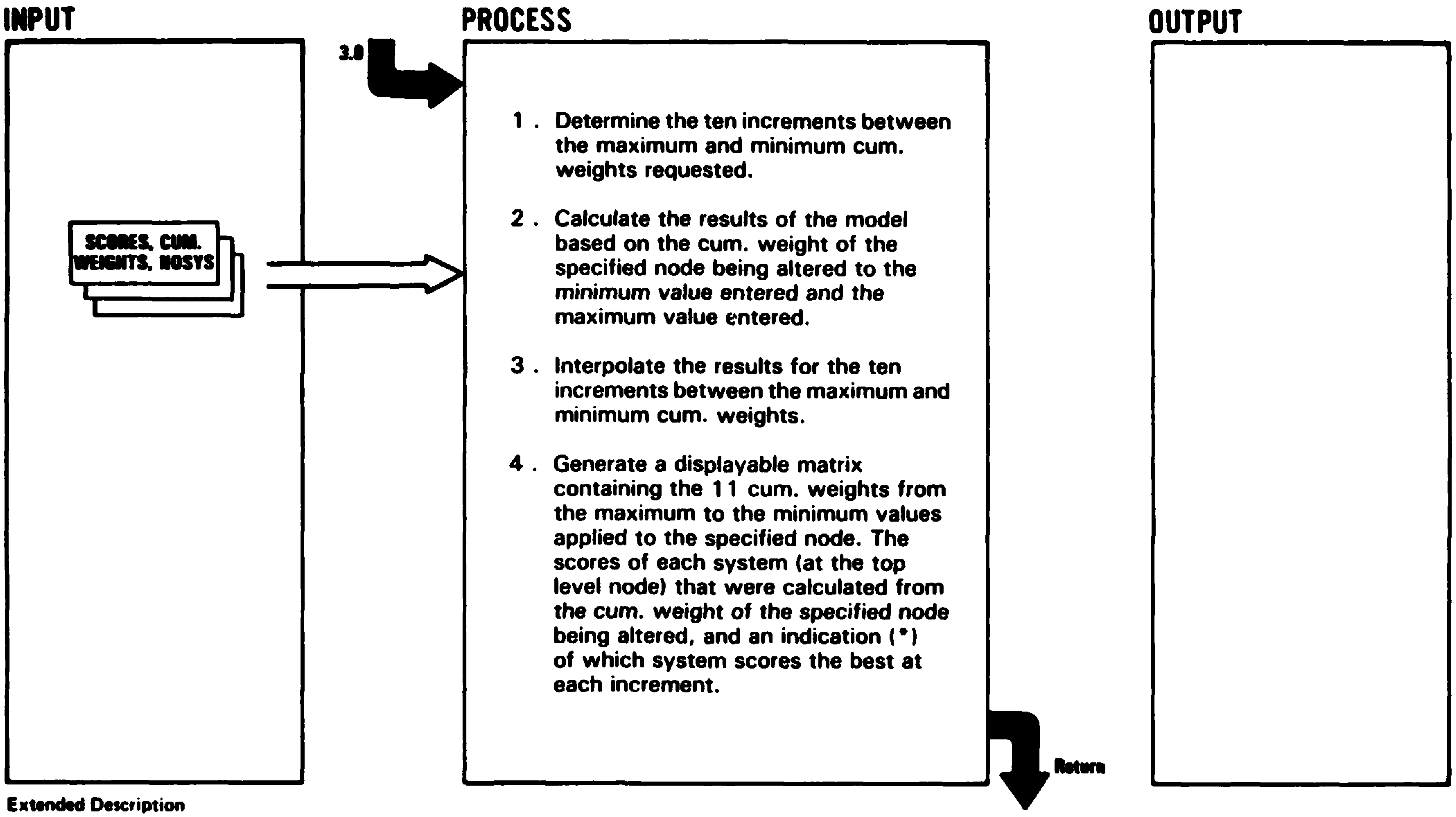
2 . Read line from display, stripping off characters vector.

3 . Strip out non numeric characters and convert to numeric vector.

4 . Set result to numeric vector with the number of elements specified by input parameter. If user entered less, pad with zeros. If user entered more, truncate.

System Program: RUN Name: CALCSENS

Diagram ID: 3.2 Description Calculate Sensitivity Values Page: ___ of ___



1 . Determine the ten increments between the maximum and minimum cum. weights requested.

2 . Calculate the results of the model based on the cum. weight of the specified node being altered to the minimum value entered and the maximum value entered.

3 . Interpolate the results for the ten increments between the maximum and minimum cum. weights.

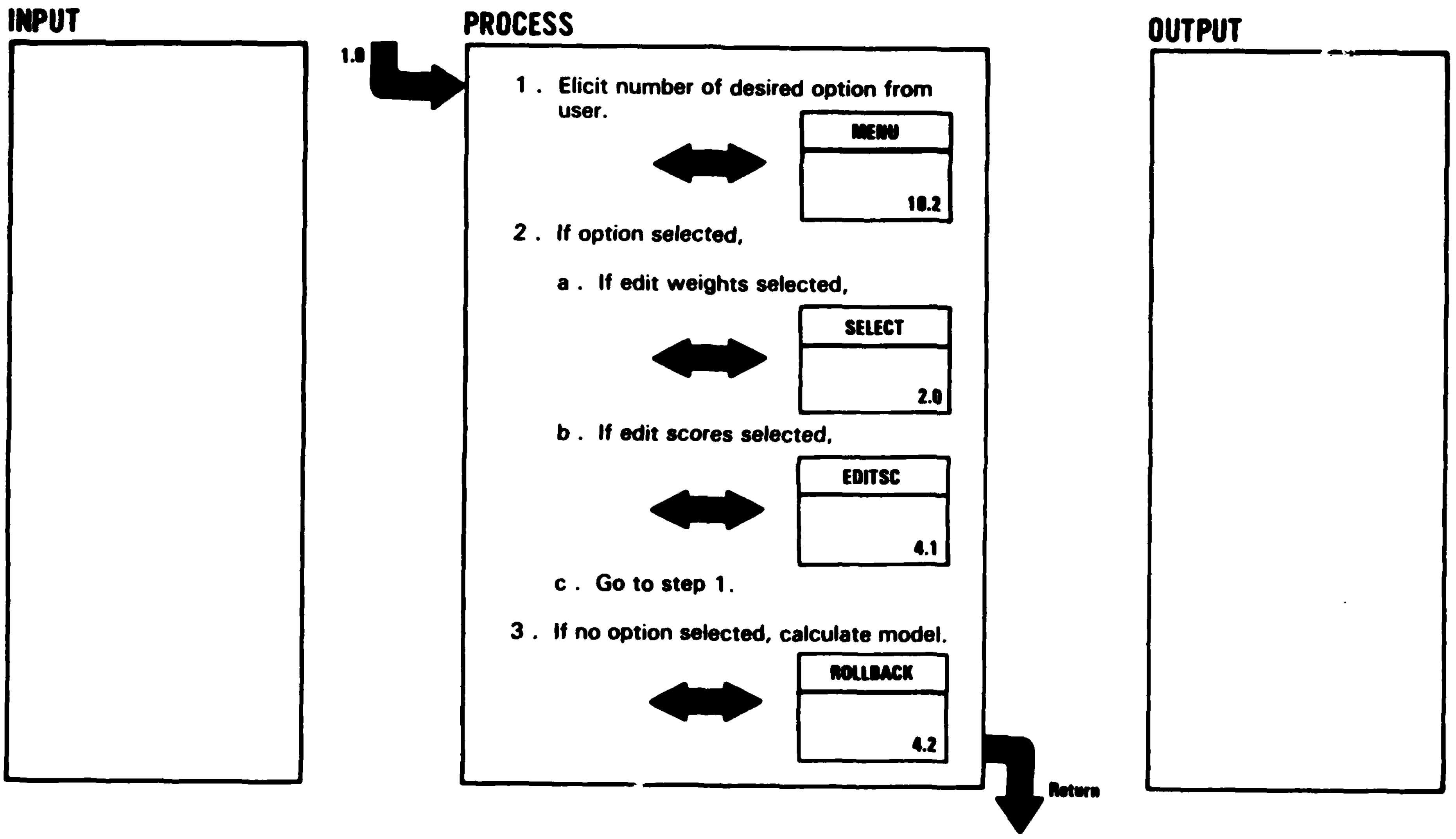
4 . Generate a displayable matrix containing the 11 cum. weights from the maximum to the minimum values applied to the specified node. The scores of each system (at the top level node) that were calculated from the cum. weight of the specified node being altered, and an indication (*) of which system scores the best at each increment.

**Extended Description**

2. When the cum. weight of the specified node is altered, the total of the cum. weights of all other nodes automatically goes up or down such that the sum remains the same. The model variable containing the cum. weights does not have to be changed from its original set of values.

System/Program: RUN Name: EDITOR

Diagram ID: 4.0 Description Edit Values Page: ___ of ___



1 . Elicit number of desired option from user.

2 . If option selected,

a . If edit weights selected,

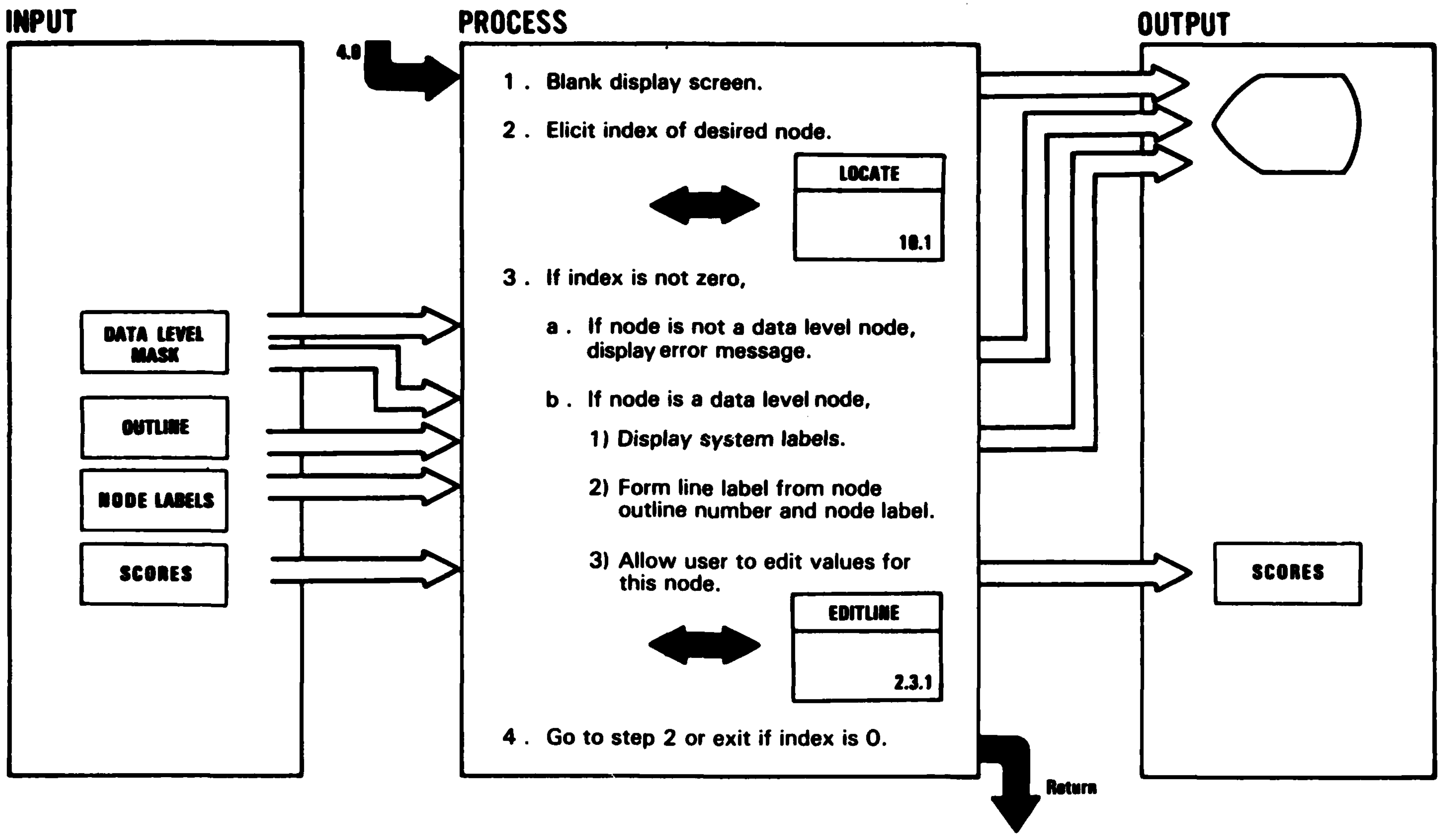
b . If edit scores selected,

c . Go to step 1.

3 . If no option selected, calculate model.

System/Program: RUN Name: EDITSC

Diagram ID: 4.1 Description Edit Data Level Node Scores Page: ___ of ___



1 . Blank display screen.

2 . Elicit index of desired node.

3 . If index is not zero,

a . If node is not a data level node, display error message.

b . If node is a data level node,

1) Display system labels.

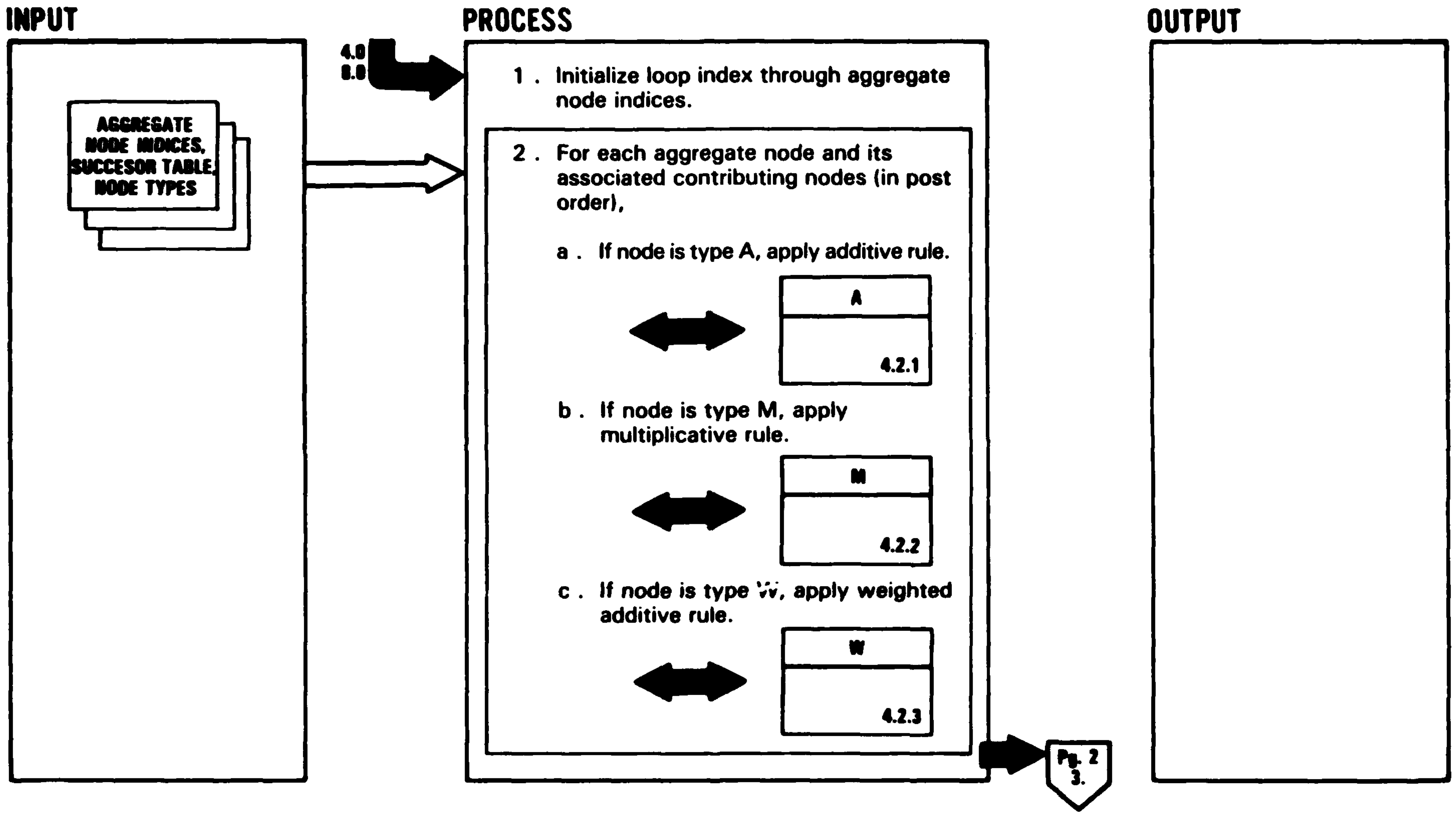
2) Form line label from node outline number and node label.

3) Allow user to edit values for this node.

4 . Go to step 2 or exit if index is 0.

System/Program: RUN Name: ROLLBACK

Diagram ID: 4.2 Description Calculate Model Page: 1 of 2



1 . Initialize loop index through aggregate node indices.

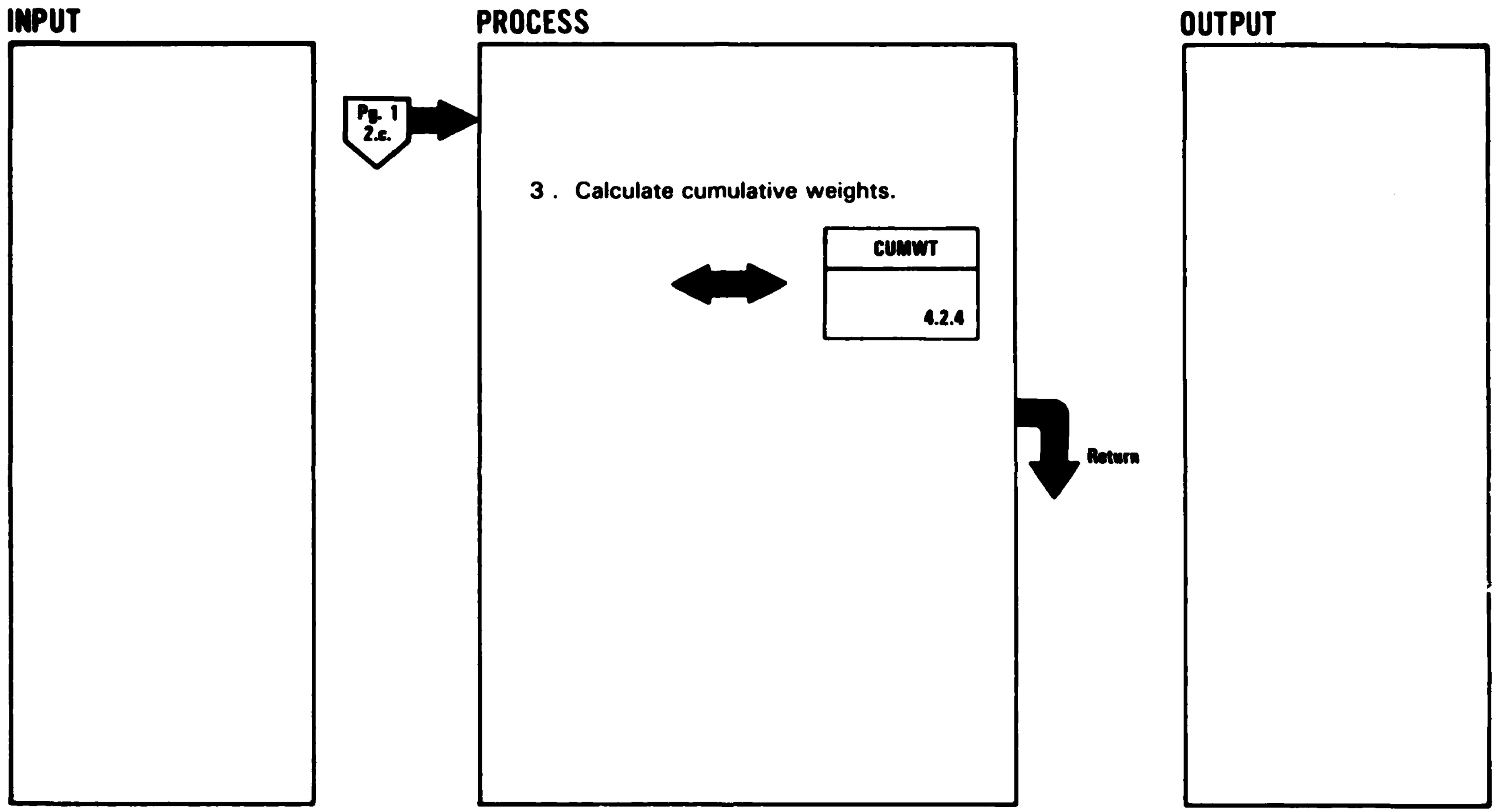
2 . For each aggregate node and its associated contributing nodes (in post order),

a . If node is type A, apply additive rule.

b . If node is type M, apply multiplicative rule.

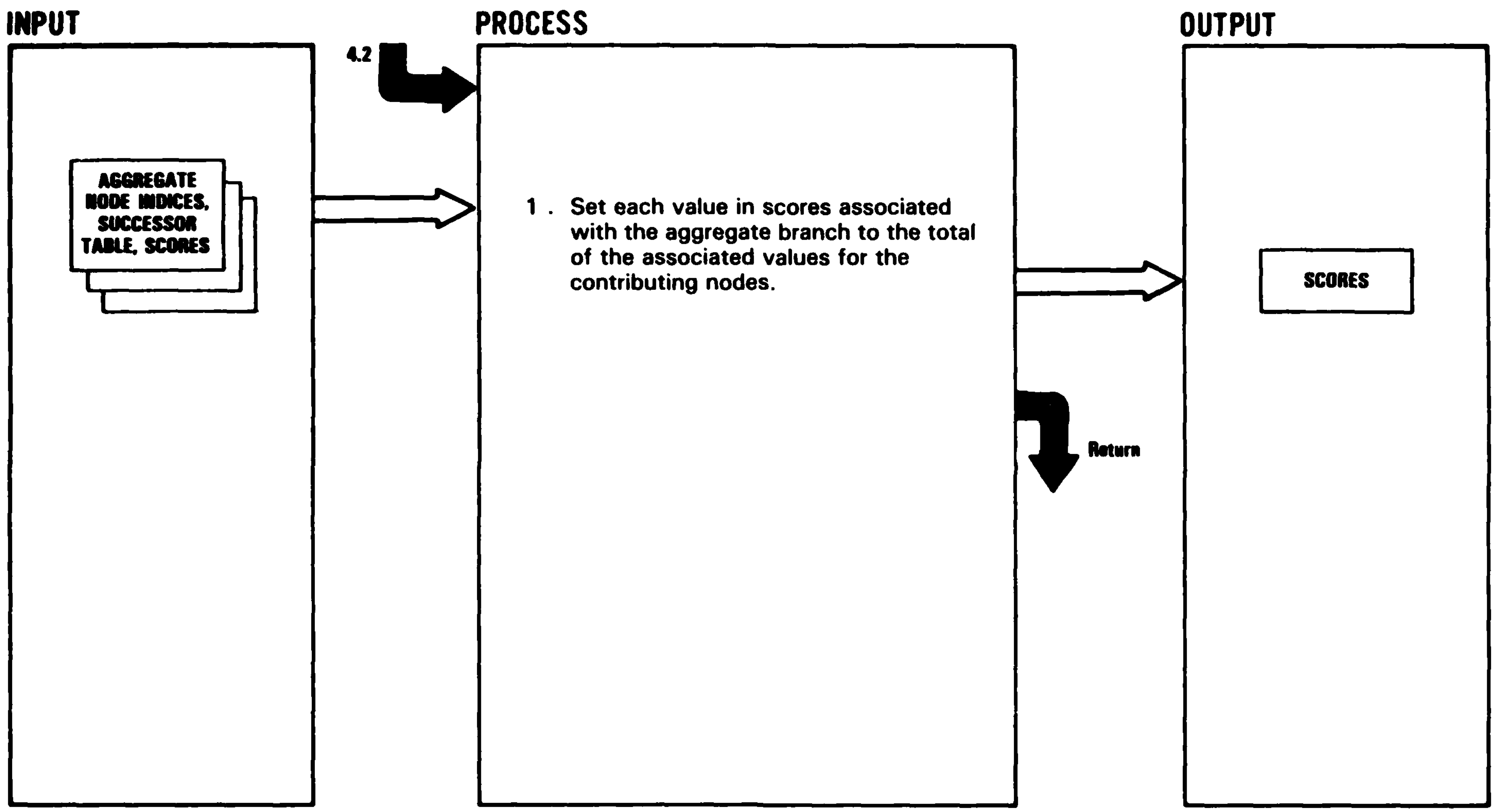
c . If node is type W, apply weighted additive rule.

System/Program: RUN Name: ROLLBACK
Diagram ID: 4.2 Description Calculate Model Page: 2 of 2



System/Program: RUN Name: A

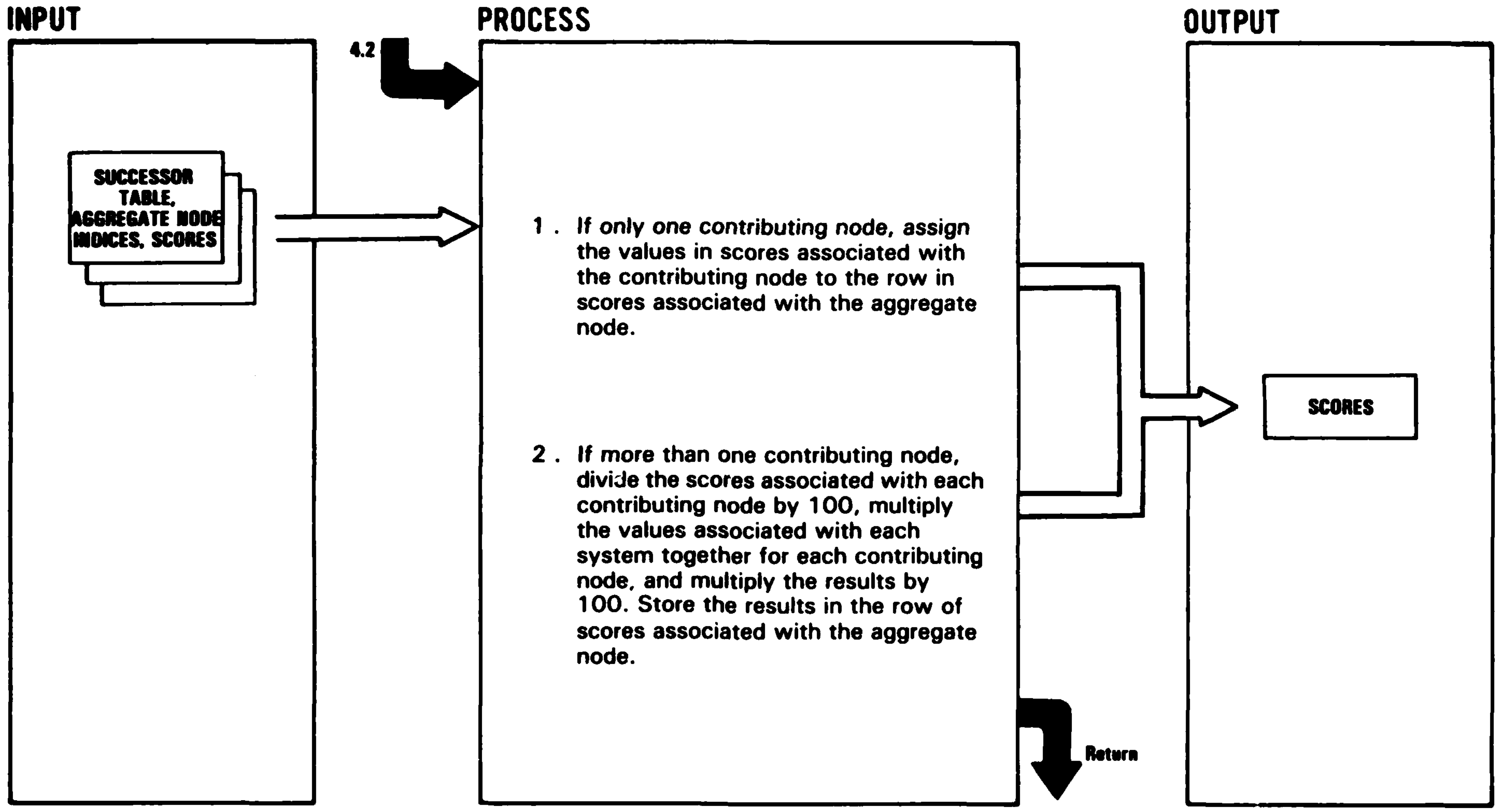
Diagram ID: 4.2.1 Description Calculate Type A Page: ___ of ___



1 . Set each value in scores associated with the aggregate branch to the total of the associated values for the contributing nodes.

System/Program: RUN Name: M

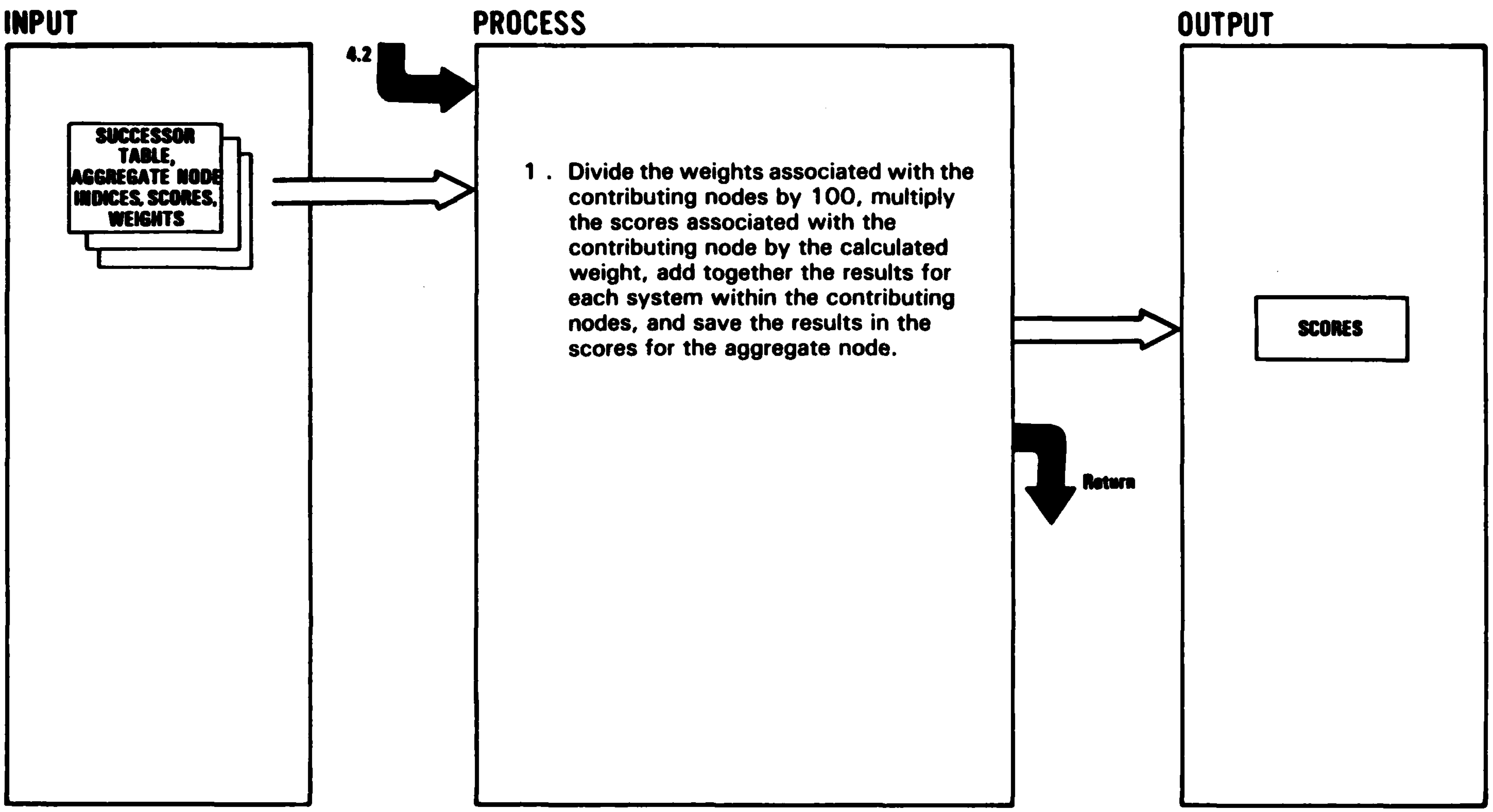
Diagram ID: 4.2.2 Description Calculate Type M Page: ___ of ___



1. If *only one* contributing node, assign the values in scores associated with the contributing node to the row in scores associated with the aggregate node.

2. If more than one contributing node, divide the scores associated with each contributing node by 100, multiply the values associated with each system together for each contributing node, and multiply the results by 100. Store the results in the row of scores associated with the aggregate node.

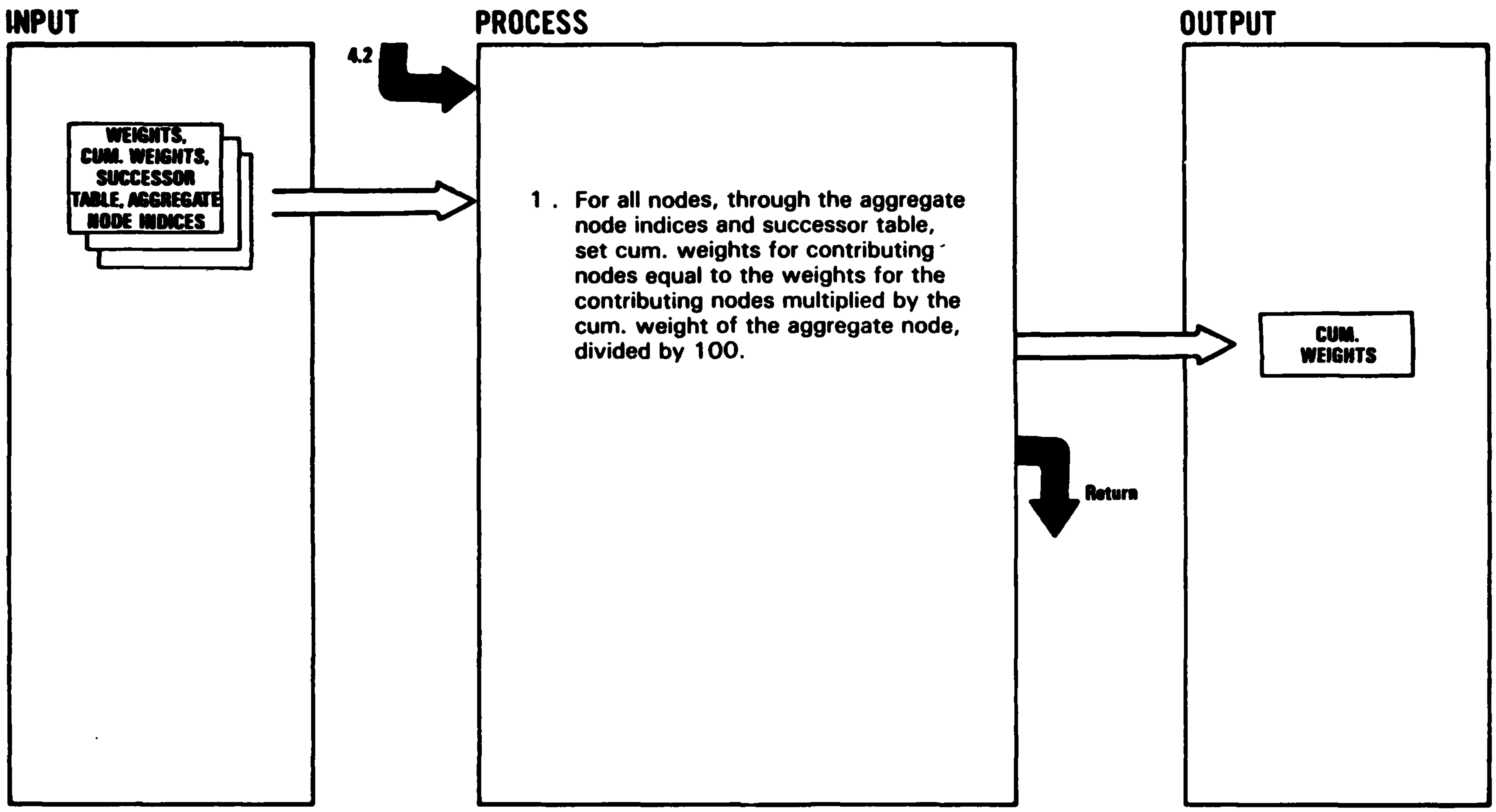
System/Program: RUN Name: W
Diagram ID: 4.2.3 Description Calculate Type W Page: of



1 . Divide the weights associated with the contributing nodes by 100, multiply the scores associated with the contributing node by the calculated weight, add together the results for each system within the contributing nodes, and save the results in the scores for the aggregate node.

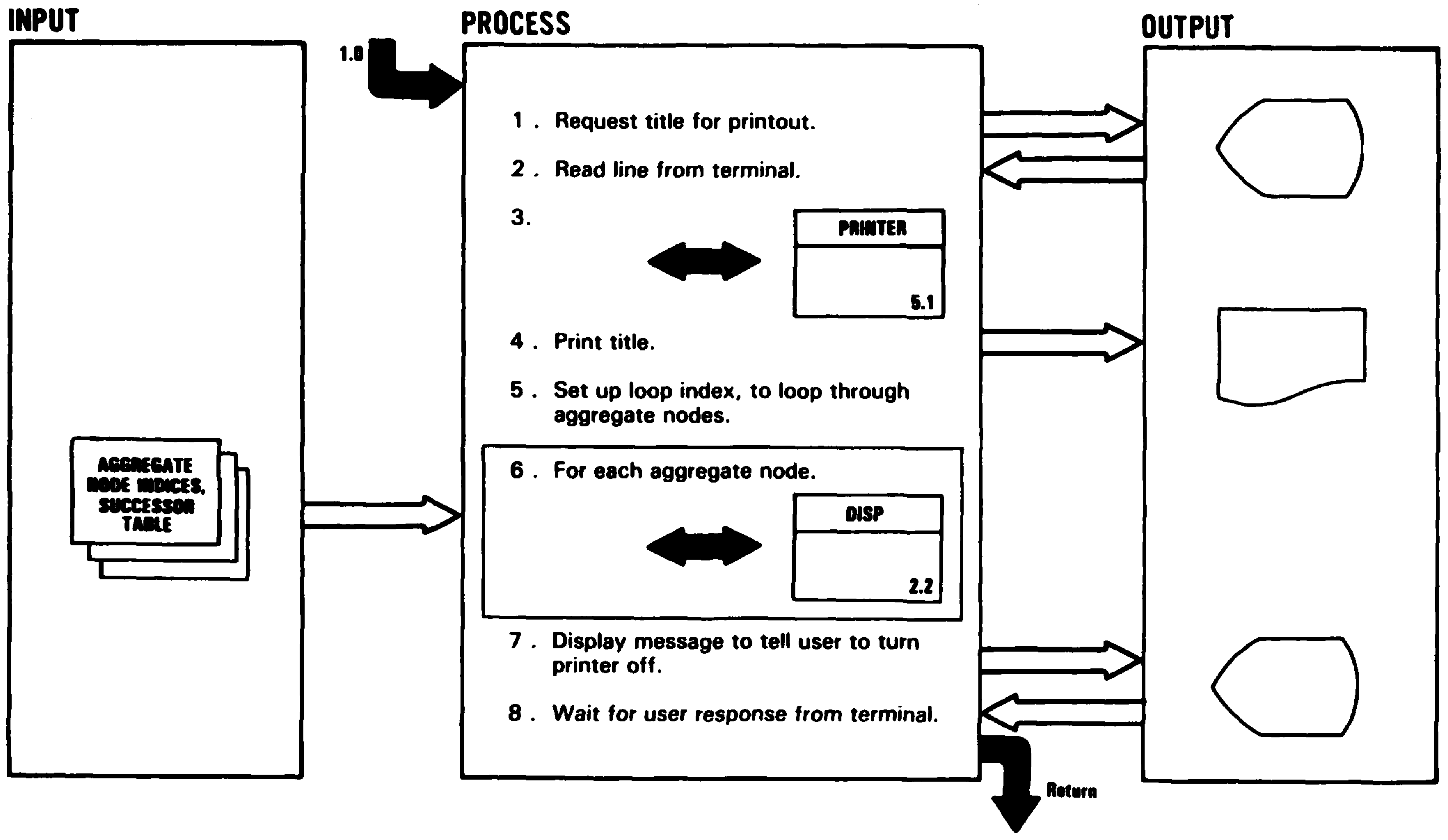
System/Program: RUN Name: CUMWT

Diagram ID: 4.2.4 Description Calculate Cumulative Weight Page: —— of ——


INPUT
PROCESS
OUTPUT
WEIGHTS, CUM. WEIGHTS, SUCCESSOR TABLE, AGGREGATE NODE INDICES
4.2
CUM. WEIGHTS
Return

1 . For all nodes, through the aggregate node indices and successor table, set cum. weights for contributing nodes equal to the weights for the contributing nodes multiplied by the cum. weight of the aggregate node, divided by 100.

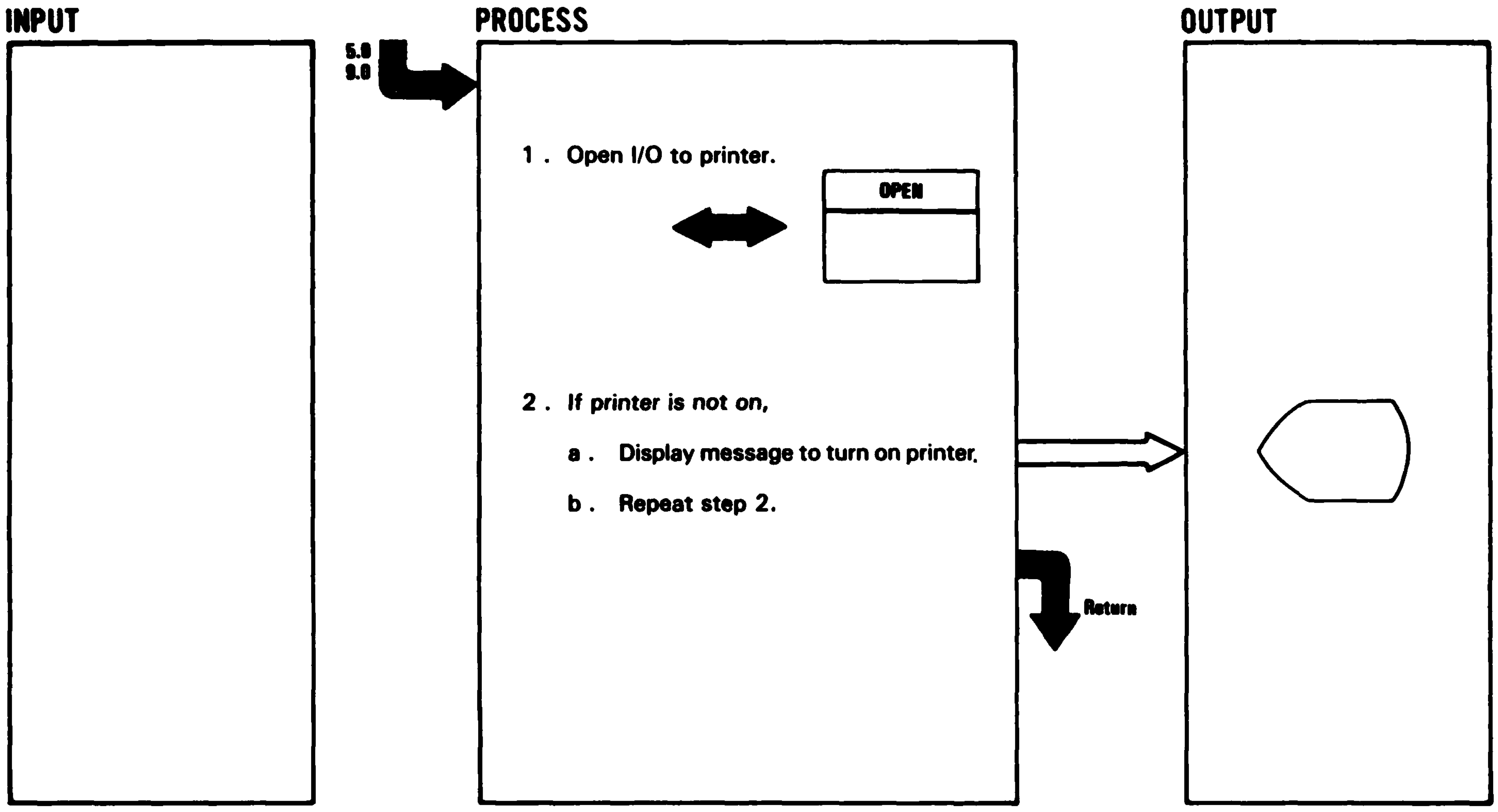
System/Program: RUN Name: SHOWALL
Diagram ID: 5.0 Description Print Results Page: ___ of ___



1 . Request title for printout.
2 . Read line from terminal.
3.
4 . Print title.
5 . Set up loop index, to loop through aggregate nodes.
6 . For each aggregate node.
7 . Display message to tell user to turn printer off.
8 . Wait for user response from terminal.

System/Program: RUN Name: PRINTER

Diagram ID: 5.1 Description Page: of



1 . Open I/O to printer.

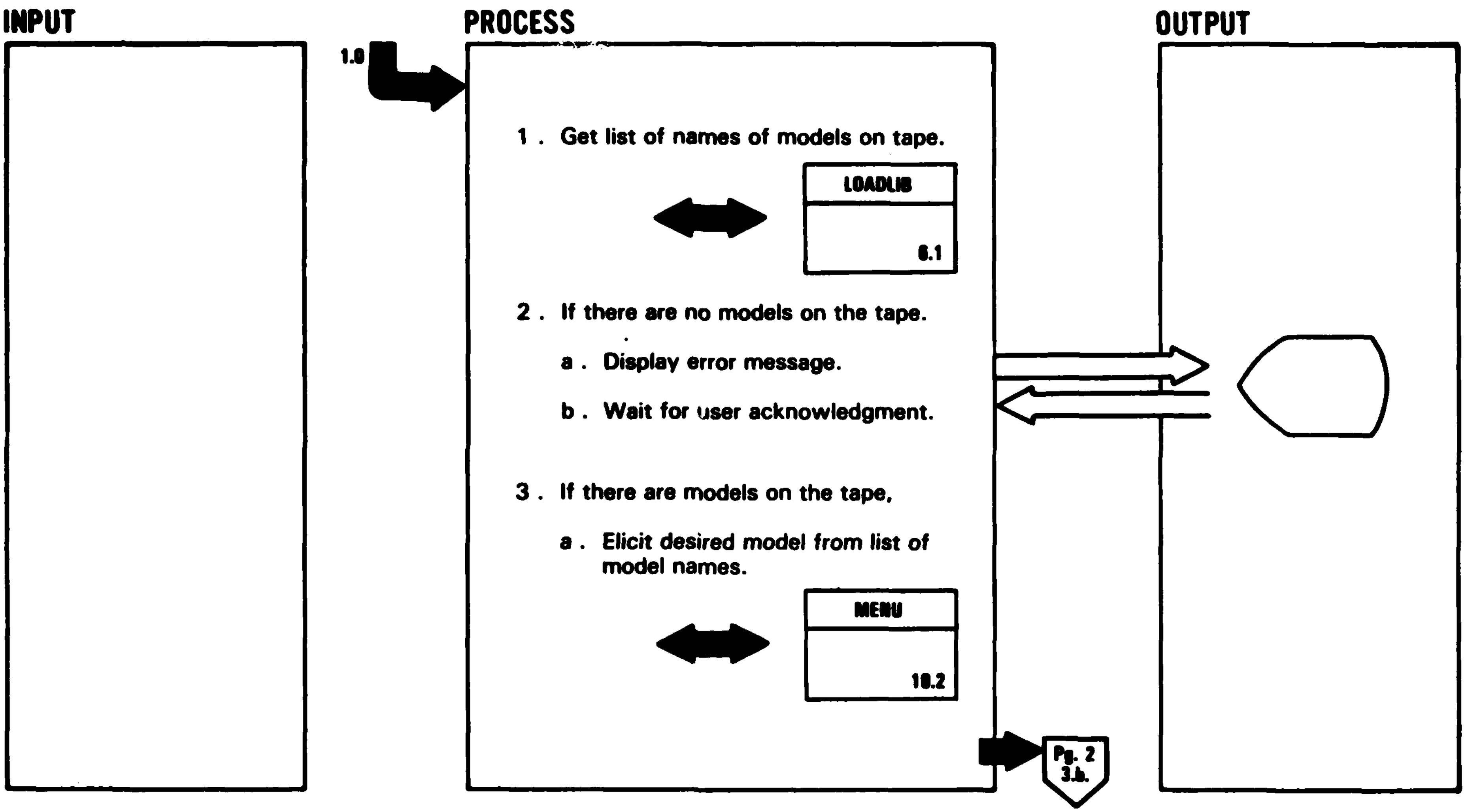
2 . If printer is not on,

a . Display message to turn on printer.

b . Repeat step 2.

System/Program: RUN Name: LOADDRIVE

Diagram ID: 6.0 Description Load Model From Tapes Page: 1 of 2



1 . Get list of names of models on tape.

2 . If there are no models on the tape.

a . Display error message.

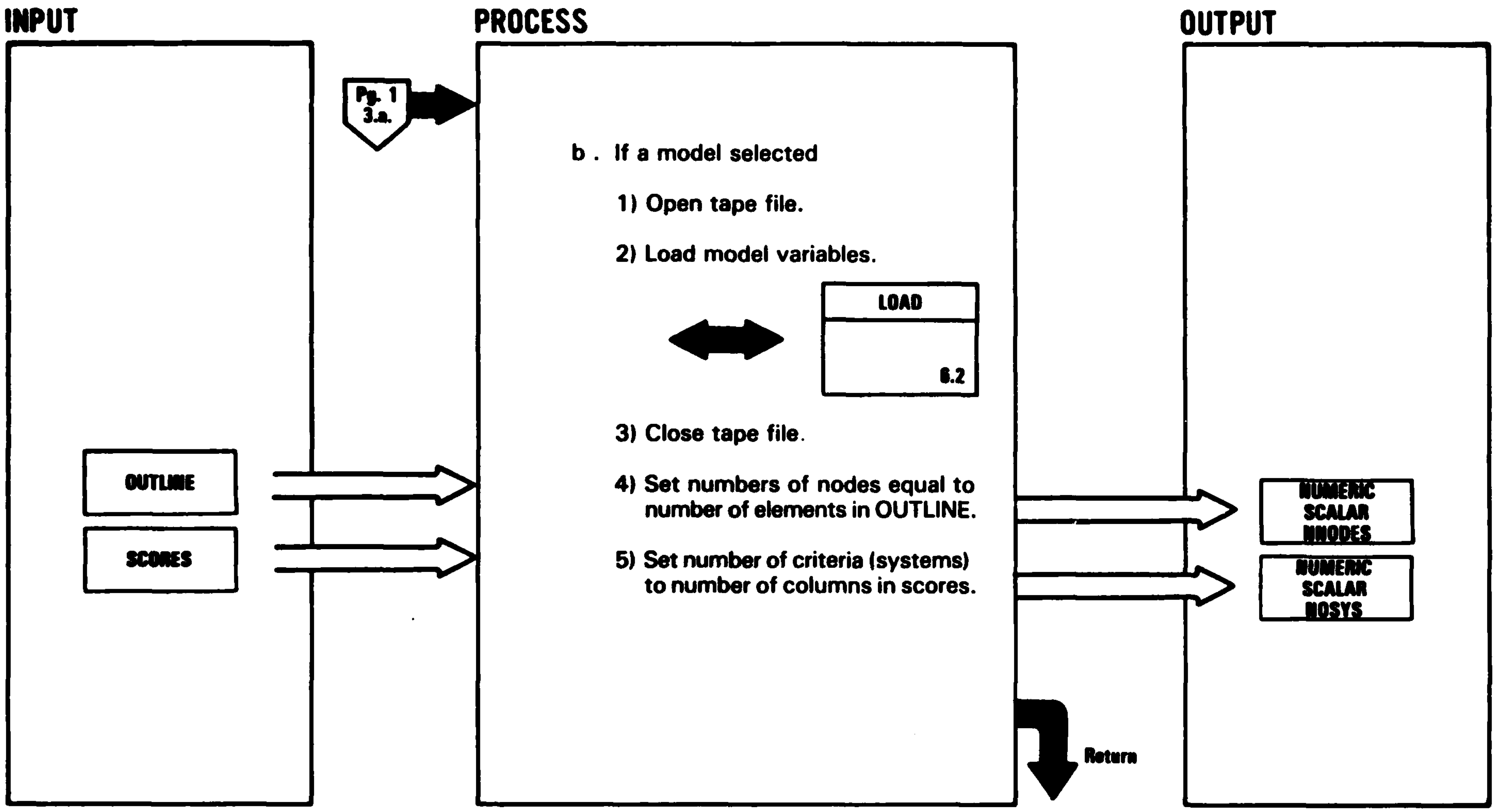
b . Wait for user acknowledgment.

3 . If there are models on the tape,

a . Elicit desired model from list of model names.

System/Program: RUN Name: LOADDRIVE

Diagram ID: 6.0 Description Load Model From Tapes Page: 2 of 2



b . If a model selected

1) Open tape file.

2) Load model variables.

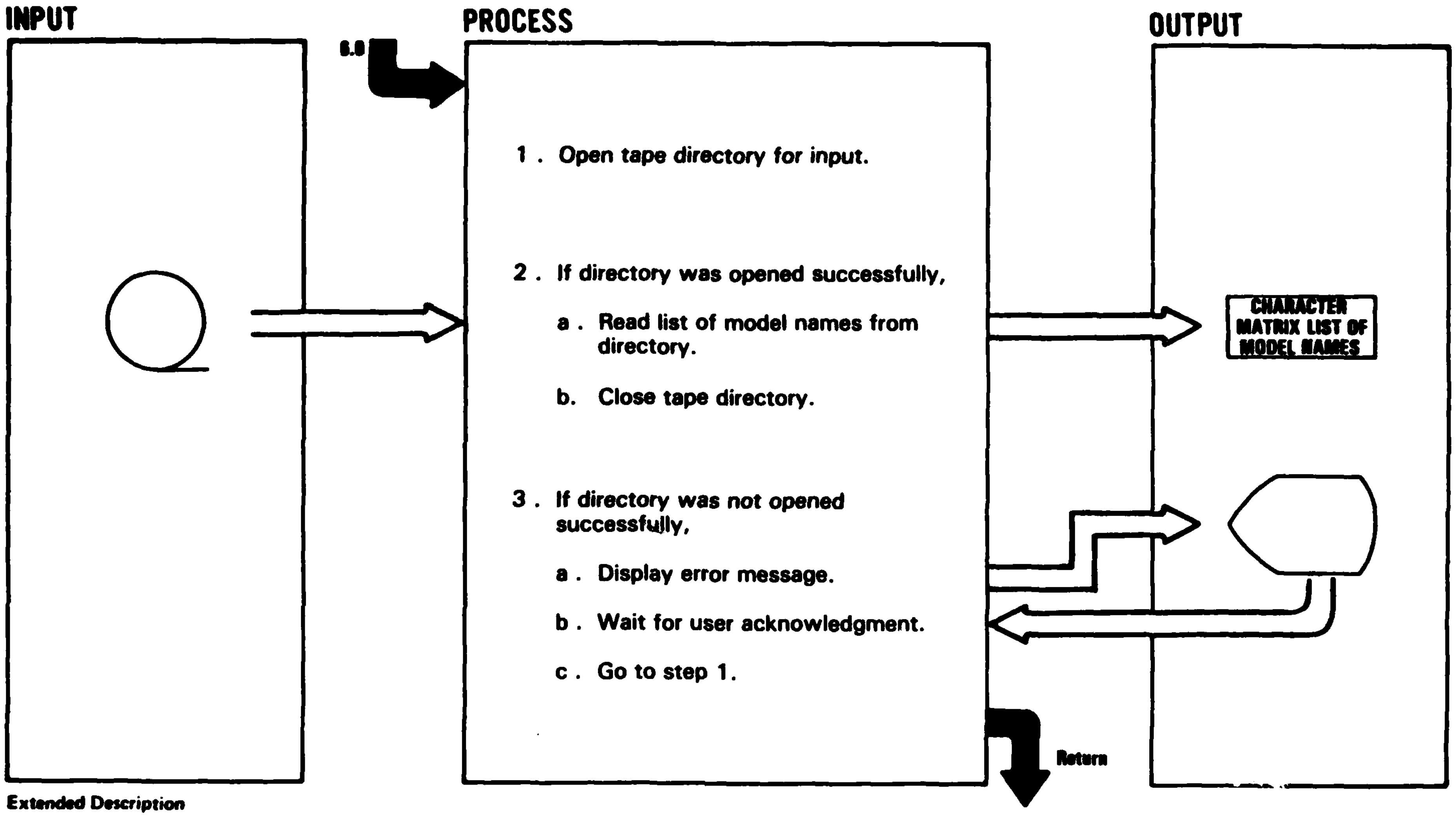
3) Close tape file.

4) Set numbers of nodes equal to number of elements in OUTLINE.

5) Set number of criteria (systems) to number of columns in scores.

System/Program: RUN Name: LOADLIB

Diagram ID: 6.1 Description Load List of Model Names Page: ___ of ___



1. Open tape directory for input.

2. If directory was opened successfully,

   a. Read list of model names from directory.

   b. Close tape directory.

3. If directory was not opened successfully,

   a. Display error message.

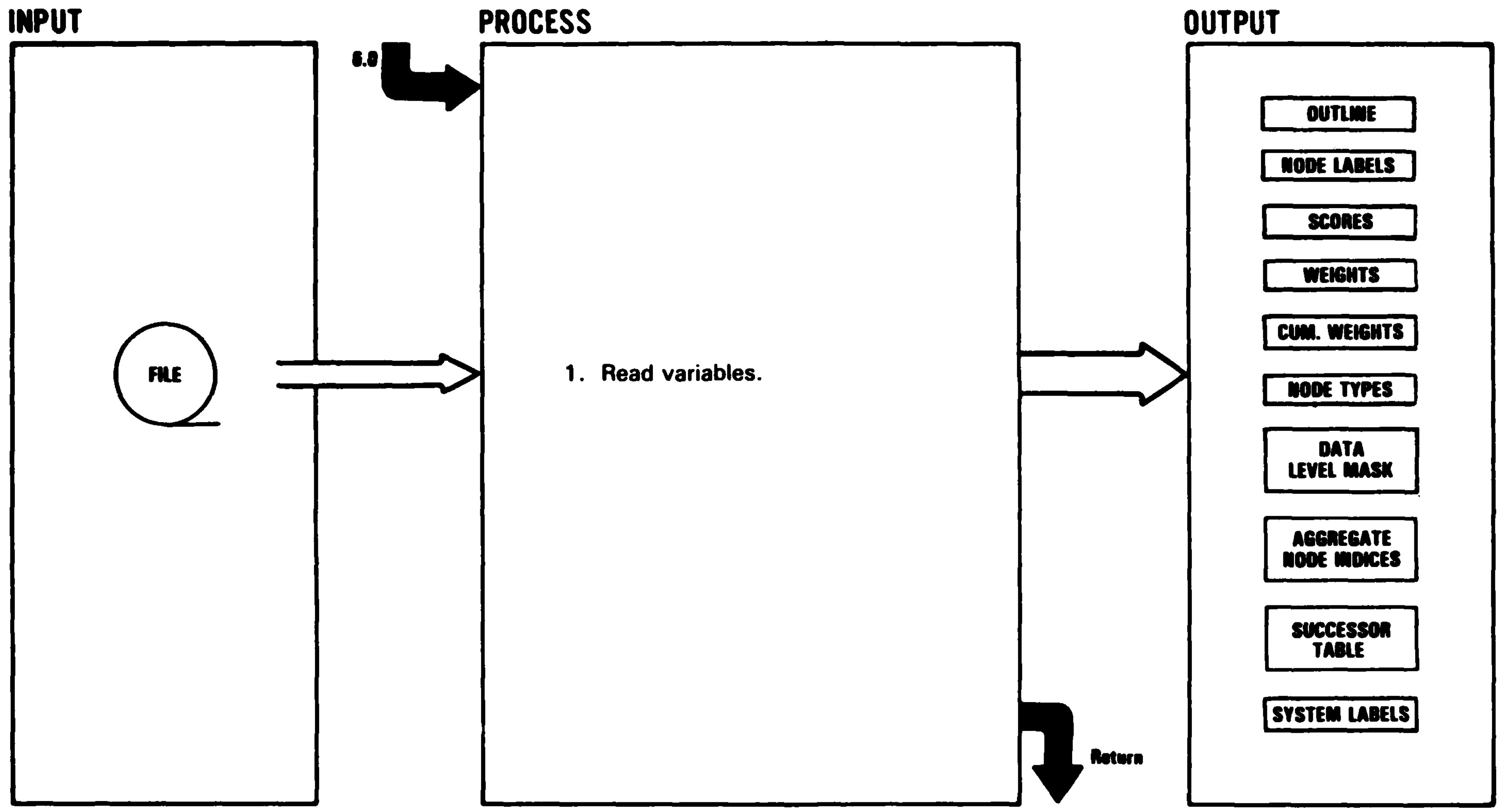
   b. Wait for user acknowledgment.

   c. Go to step 1.

**Extended Description**

Position of model names within list indicates where models are stored on tape.

System/Program: RUN Name: LOAD
Diagram ID: 6.2 Description Load Model Variables Page: 1 of 2



**Extended Description**

1. The OUTLINE TABLE contains an element for each node in the model, sorted in increasing numerical sequence order. The value is an encoded representation of the node outline number supplied for a node when the model structure is created.

2. The NODE LABELS contain descriptions (one per node in the same order as the outline table) of nodes that are supplied when the model structure is created.

3. SCORES is a numeric array which contains a set of values for each node of the structure. Each set of values consists of one number per system defined in the model.

4. WEIGHTS is a numeric vector containing the relative-importance values assigned to each node in the model structure. The elements must appear in the same order as the associated outline numbers. When a model structure is created, the vector is null or contains zeros.

5. For each element in the node outline table, there is an associated element in the CUMULATIVE WEIGHTS vector. The vector will contain the percentage of importance with respect to the entire model when all WEIGHTS have been entered.

6. The NODE TYPES are indicators of the type of calculation that is to be used in assessing SCORES and WEIGHTS.

| | | |
|---|---|---|
| System/Program: RUN | Name: LOAD | |
| Diagram ID: 6.2 | Description Load Model Variables | Page: 2 of 2 |

**INPUT**

**PROCESS**

**OUTPUT**

**Extended Description**

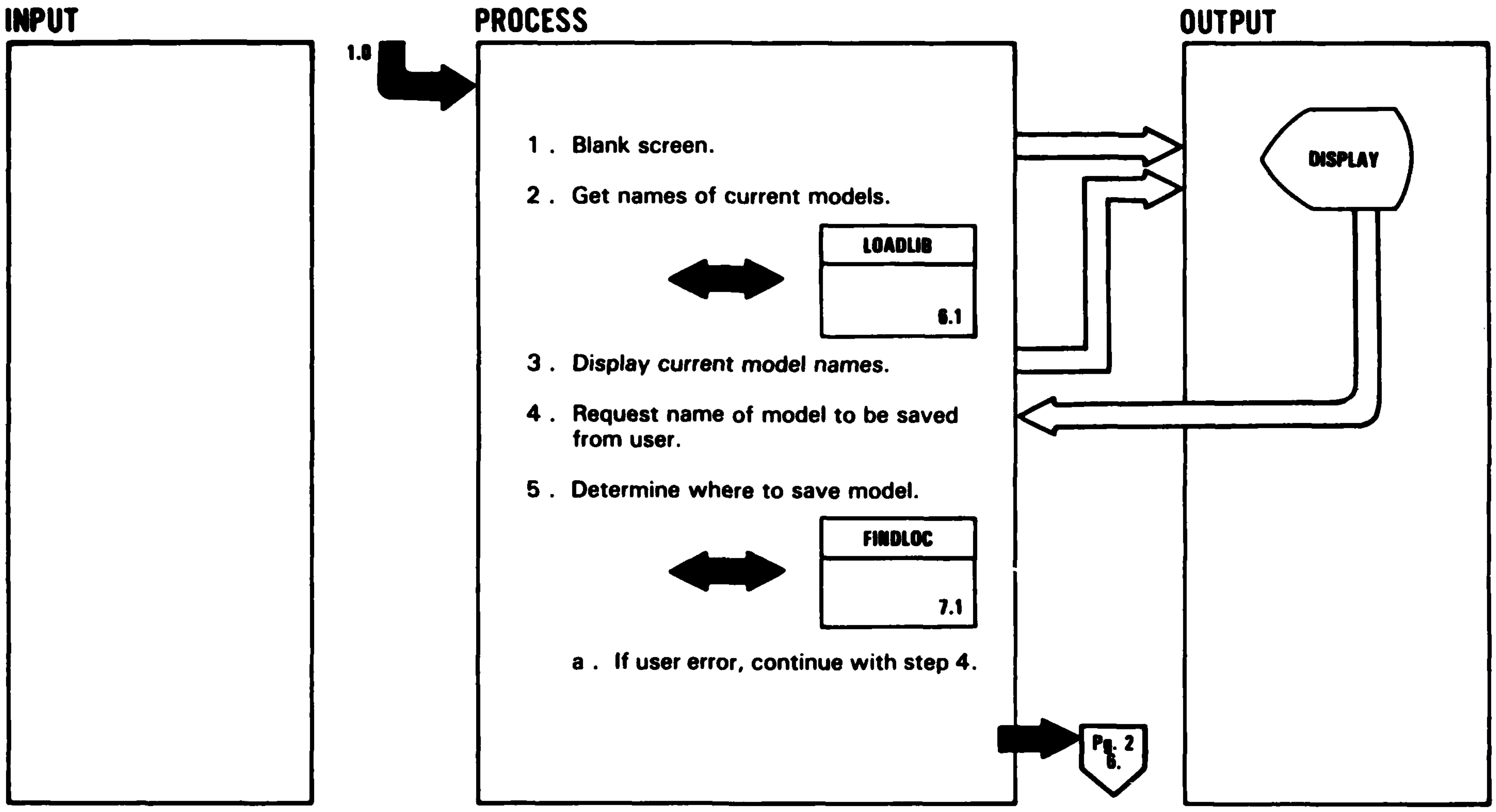
7. The DATA LEVEL MASK indicates which nodes are at the data level (bottom level) versus the nodes that are aggregate or non-bottom-level nodes.

8. The AGGREGATE NODE INDICES contain the sequence number of elements in the model variables which correspond to only the aggregate nodes. An Aggregate node is a node which has one or more subsequent nodes contributing to it.

9. The SUCCESSOR TABLE is an array which contains, for each aggregate node, the set of indices of nodes which contribute to a node.

10. The SYSTEMS LABELS contain the user-specified character descriptions of the systems being evaluated.

System/Program: RUN  Name: SAVEDRIVE
Diagram ID: 7.0  Description  Page: 1 of 2



1 . Blank screen.

2 . Get names of current models.

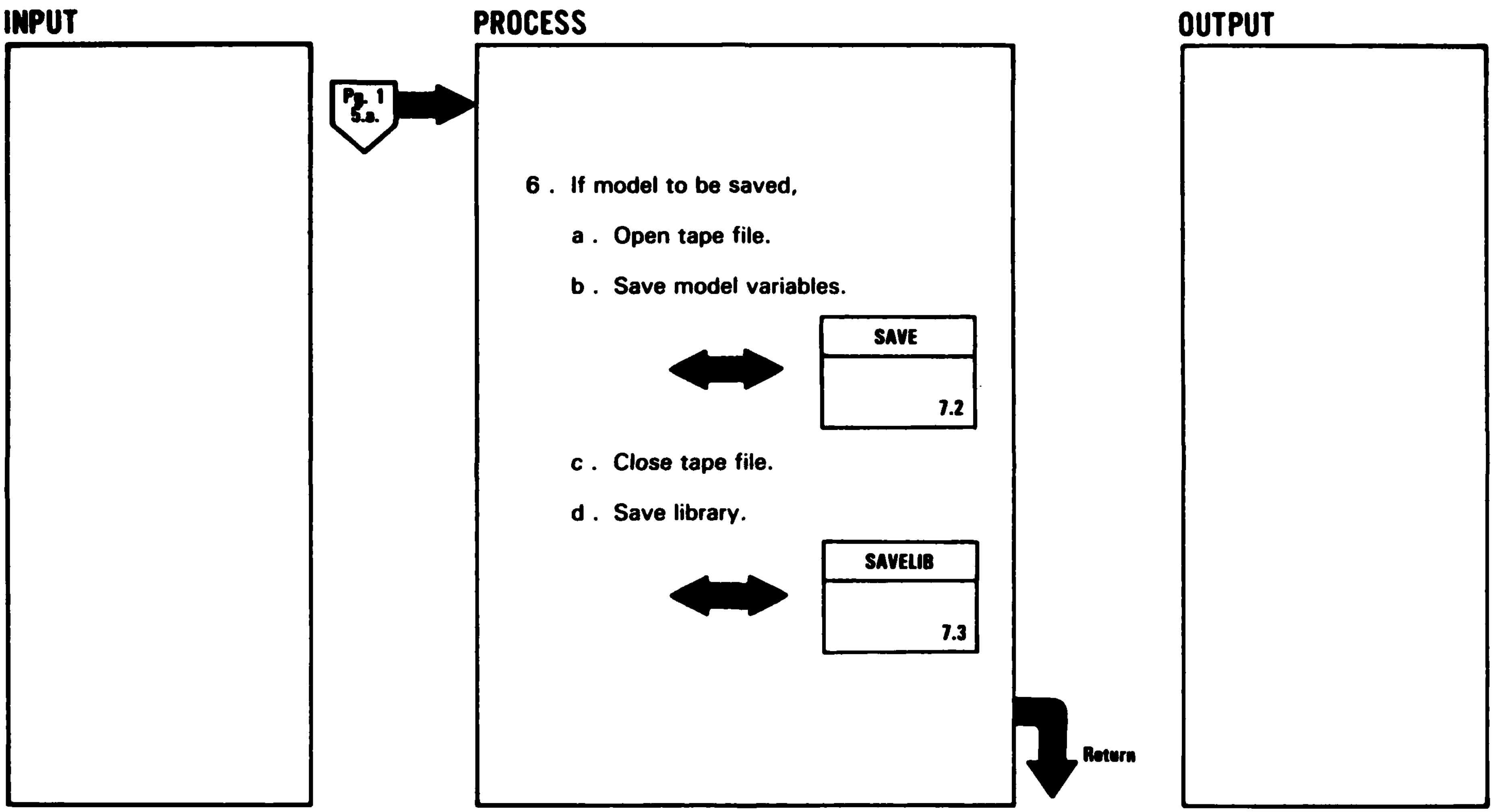
3 . Display current model names.

4 . Request name of model to be saved from user.

5 . Determine where to save model.

a . If user error, continue with step 4.

| | | |
|---|---|---|
| System/Program: RUN | Name: SAVEDRIVE | |
| Diagram ID: 7.0 | Description | Page: 2 of 2 |



6 . If model to be saved,

a . Open tape file.

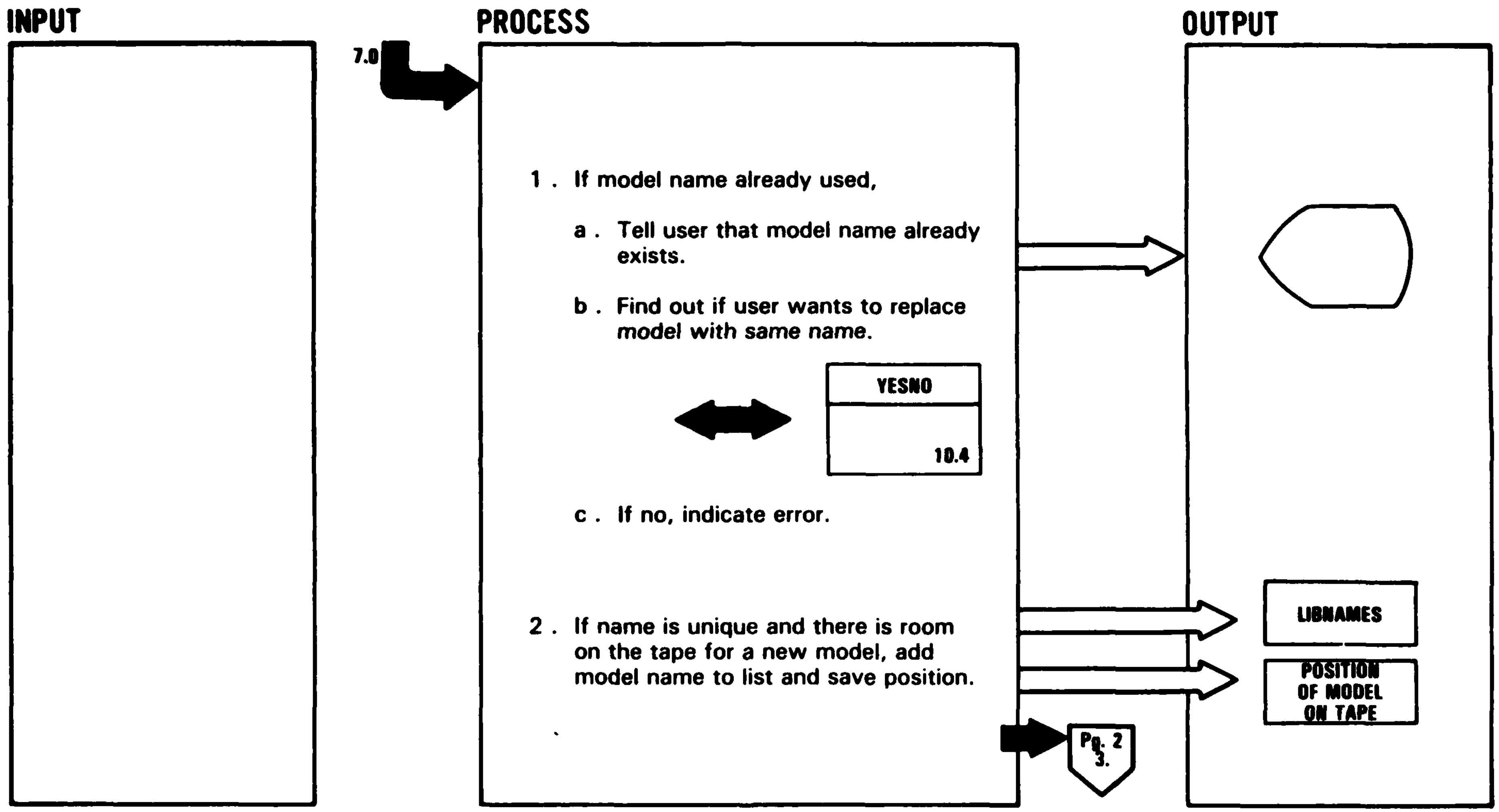
b . Save model variables.

c . Close tape file.

d . Save library.

System/Program: RUN Name: FINDLOC

Diagram ID: 7.1 Description Determine Where to Save Model Page: 1 of 2



INPUT

PROCESS

1 . If model name already used,

a . Tell user that model name already exists.

b . Find out if user wants to replace *model* with *same name*.

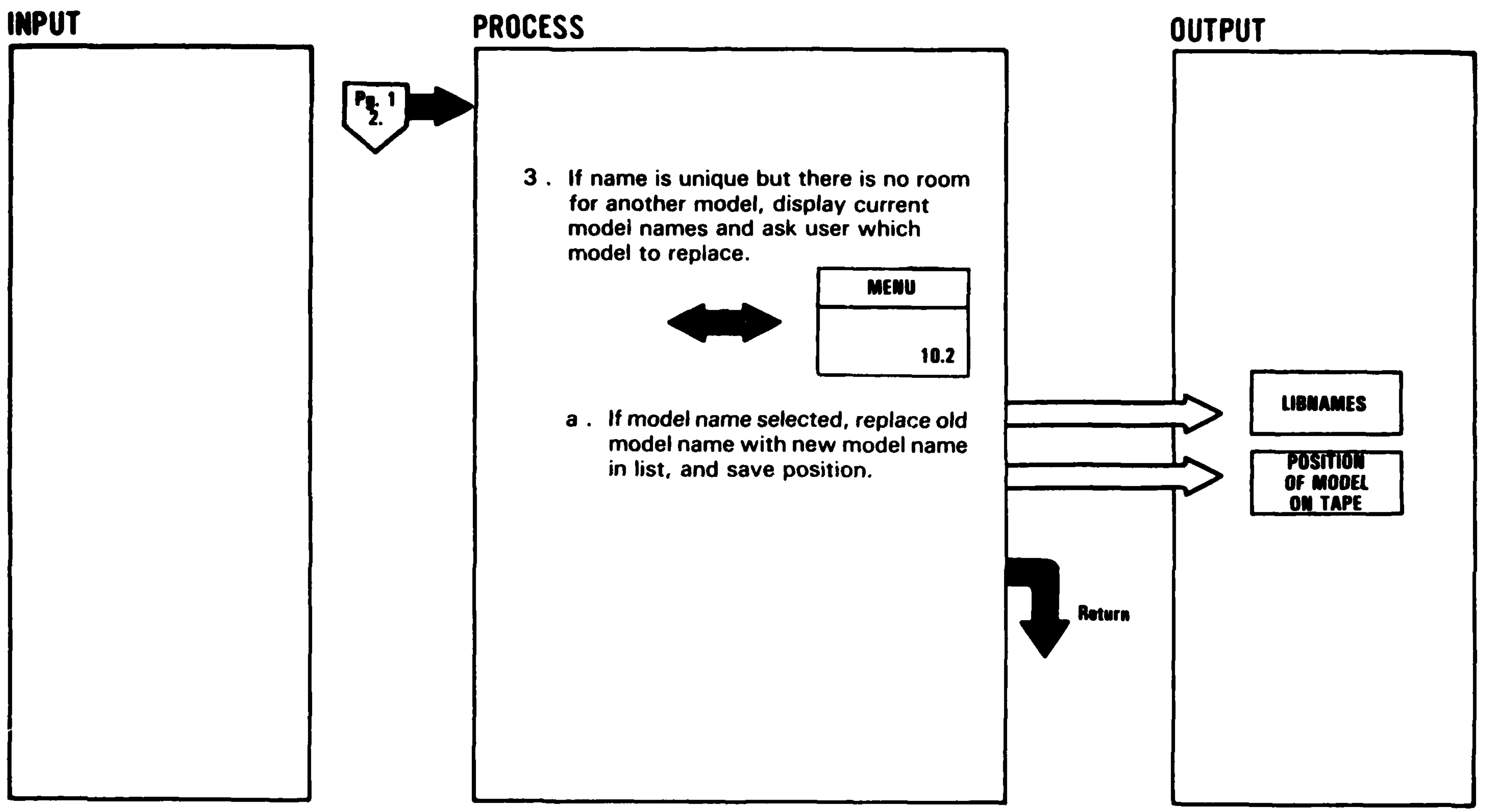
c . If no, indicate error.

2 . If name is unique and there is room on the tape for a new model, add model name to list and save position.

OUTPUT

System/Program: RUN Name: FINDLOC

Diagram ID: 7.1 Description Determine Where to Save Model Page: 2 of 2

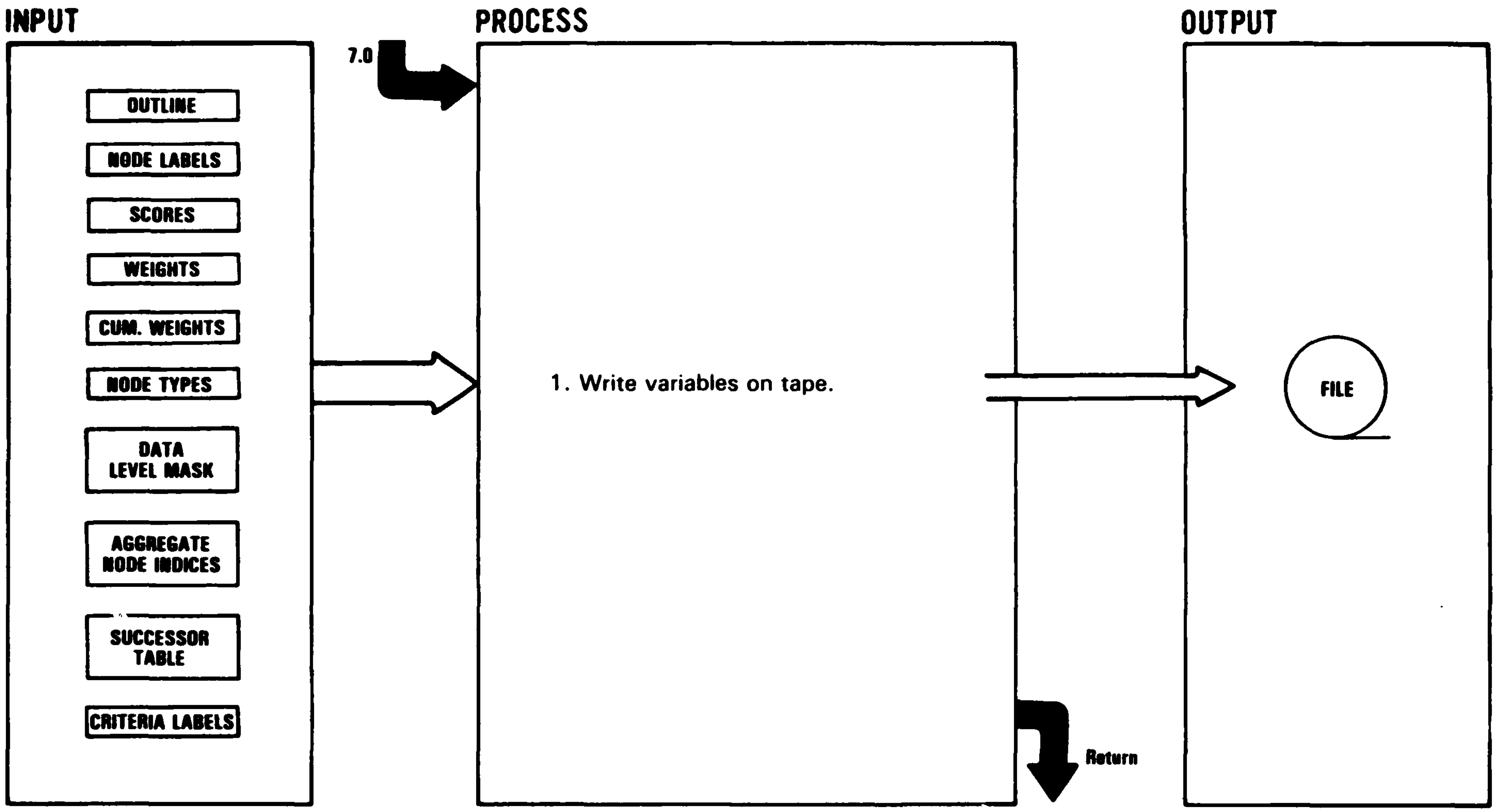


3 . If name is unique but there is no room for another model, display current model names and ask user which model to replace.

a . If model name selected, replace old model name with new model name in list, and save position.

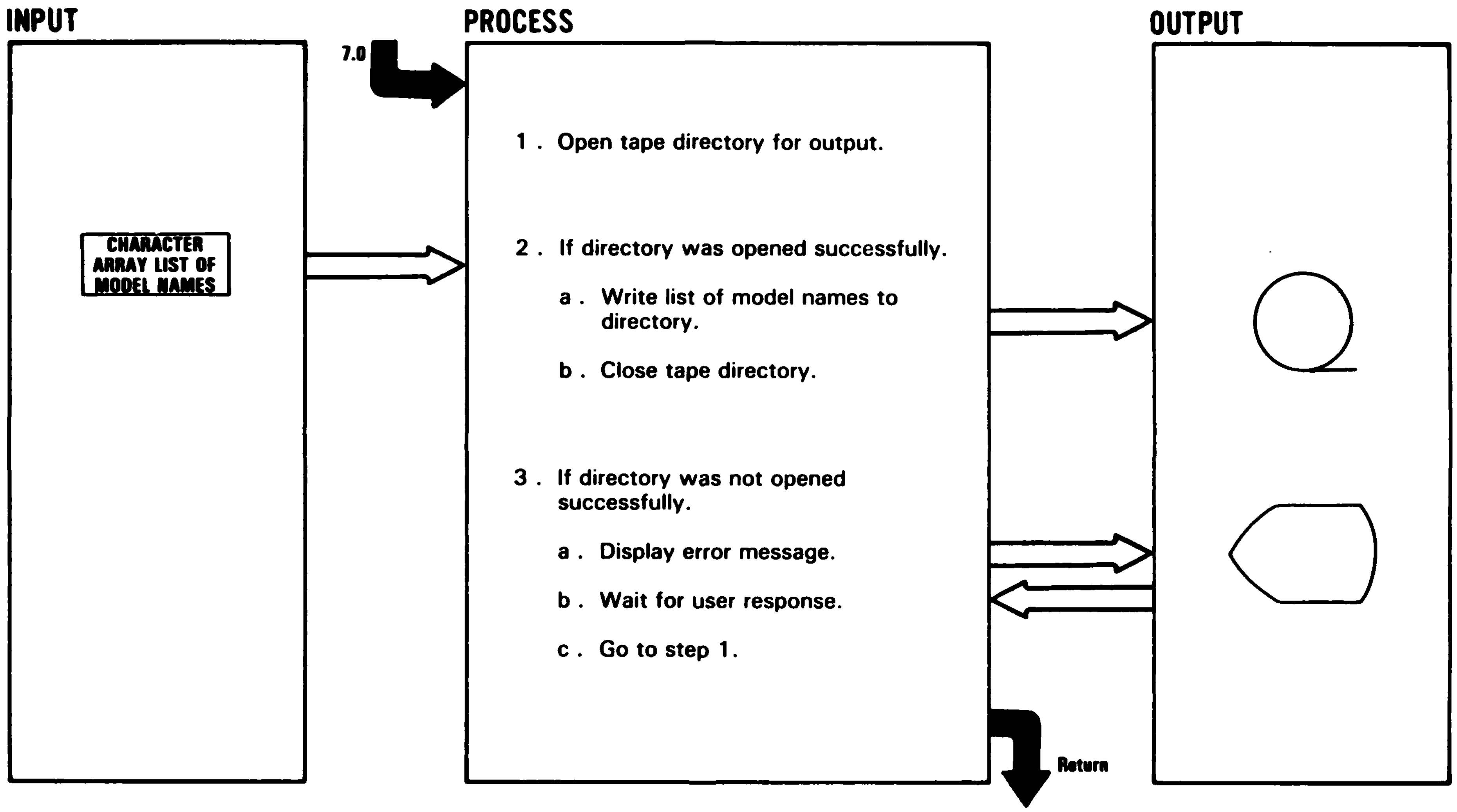
System/Program: RUN Name: SAVE

Diagram ID: 7.2 Description Save Model Variables on Tape Page: ____ of ____



System/Program: RUN Name: SAVLIB

Diagram ID: 7.3 Description Save List of Model Names Page: ___ of ___



1 . Open tape directory for output.

2 . If directory was opened successfully.

a . Write list of model names to directory.

b . Close tape directory.

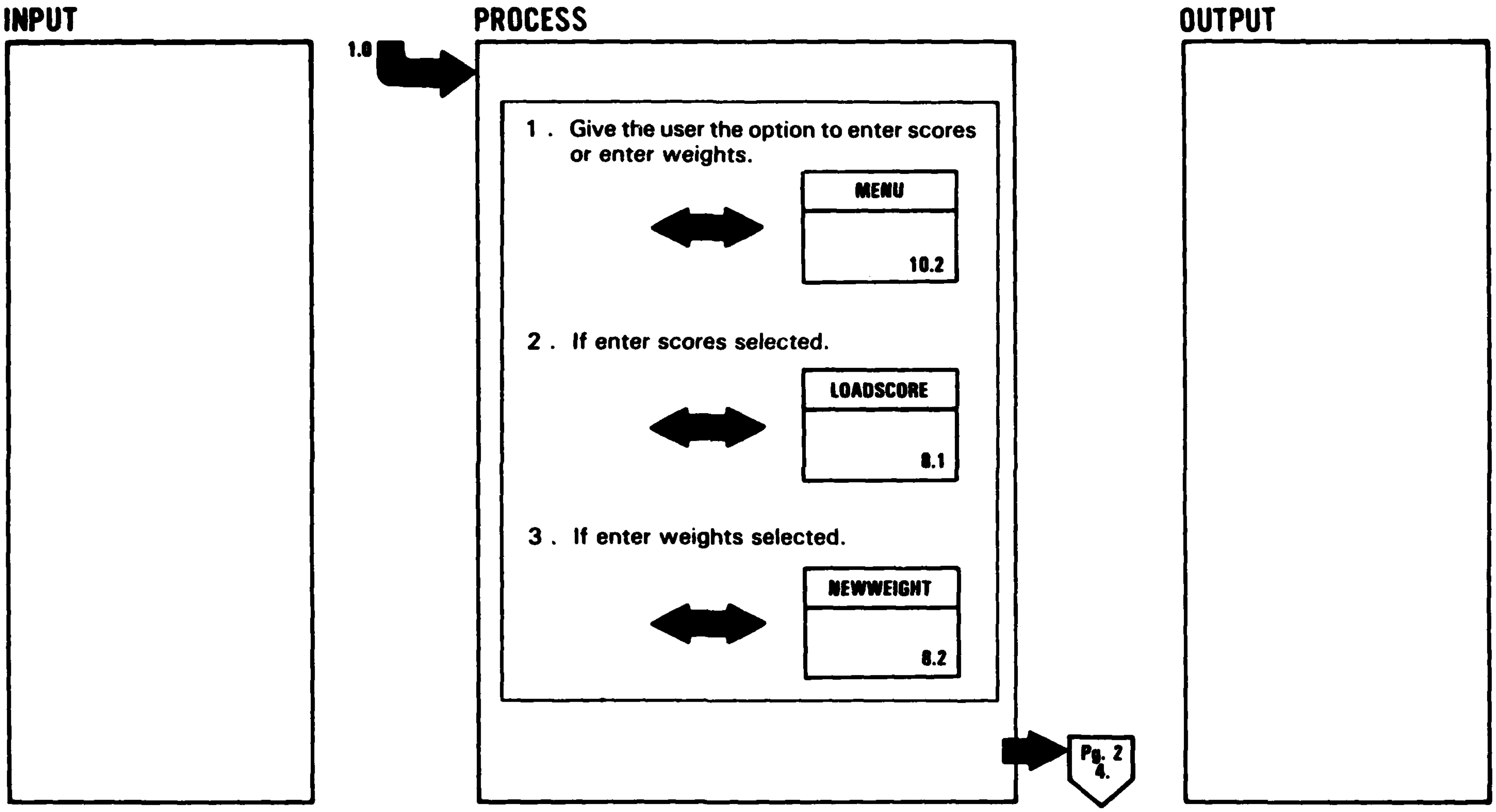
3 . If directory was not opened successfully.

a . Display error message.

b . Wait for user response.

c . Go to step 1.

| | |
|---|---|
| System/Program: RUN | Name: NEWDATA |
| Diagram ID: 8.0 Description | Page: 1 of 2 |



INPUT

PROCESS

1 . Give the user the option to enter scores or enter weights.

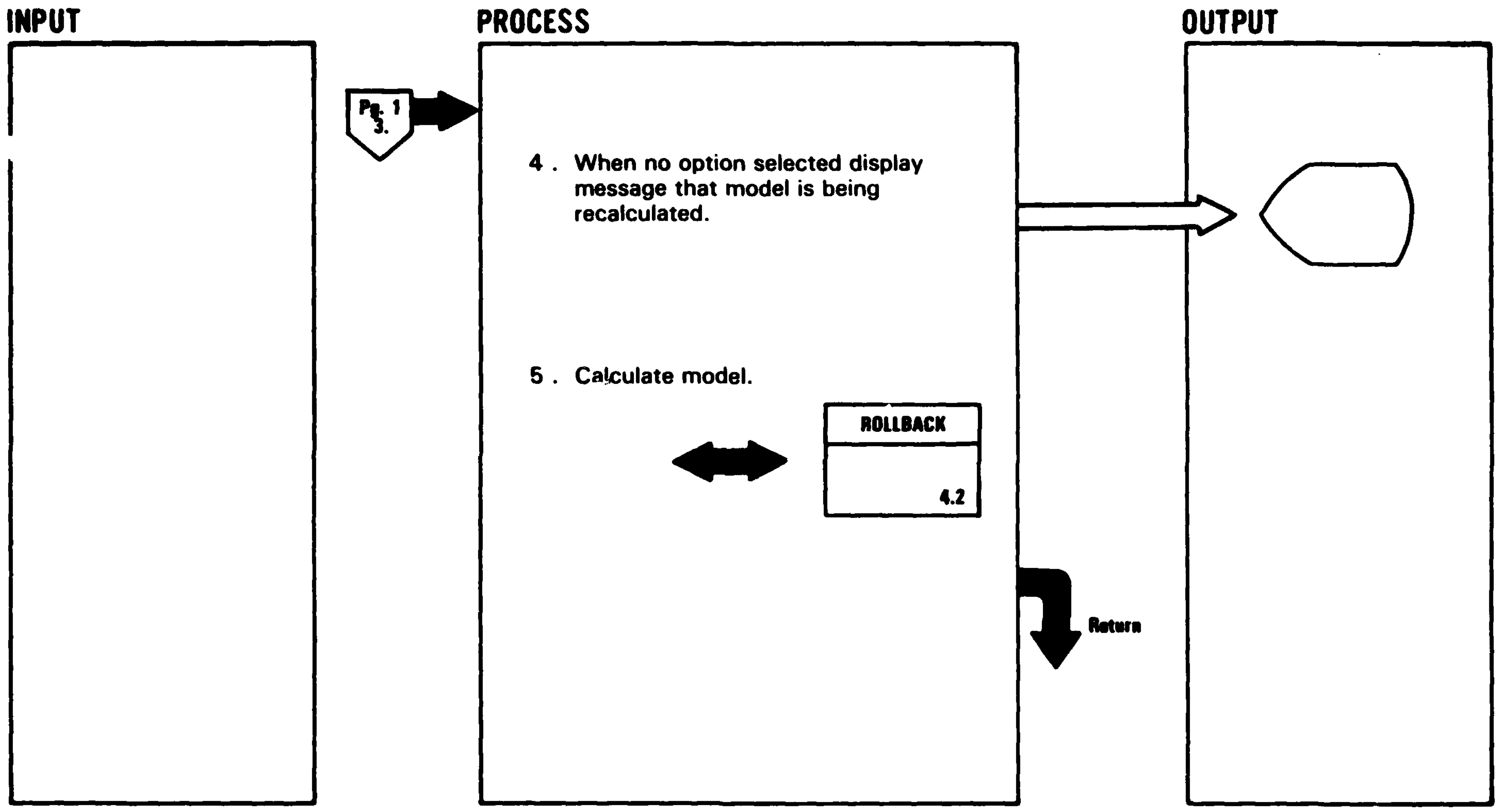
2 . If enter scores selected.

3 . If enter weights selected.

OUTPUT

System/Program: RUN Name: NEWDATA

Diagram ID: 8.0 Description Page: 2 of 2

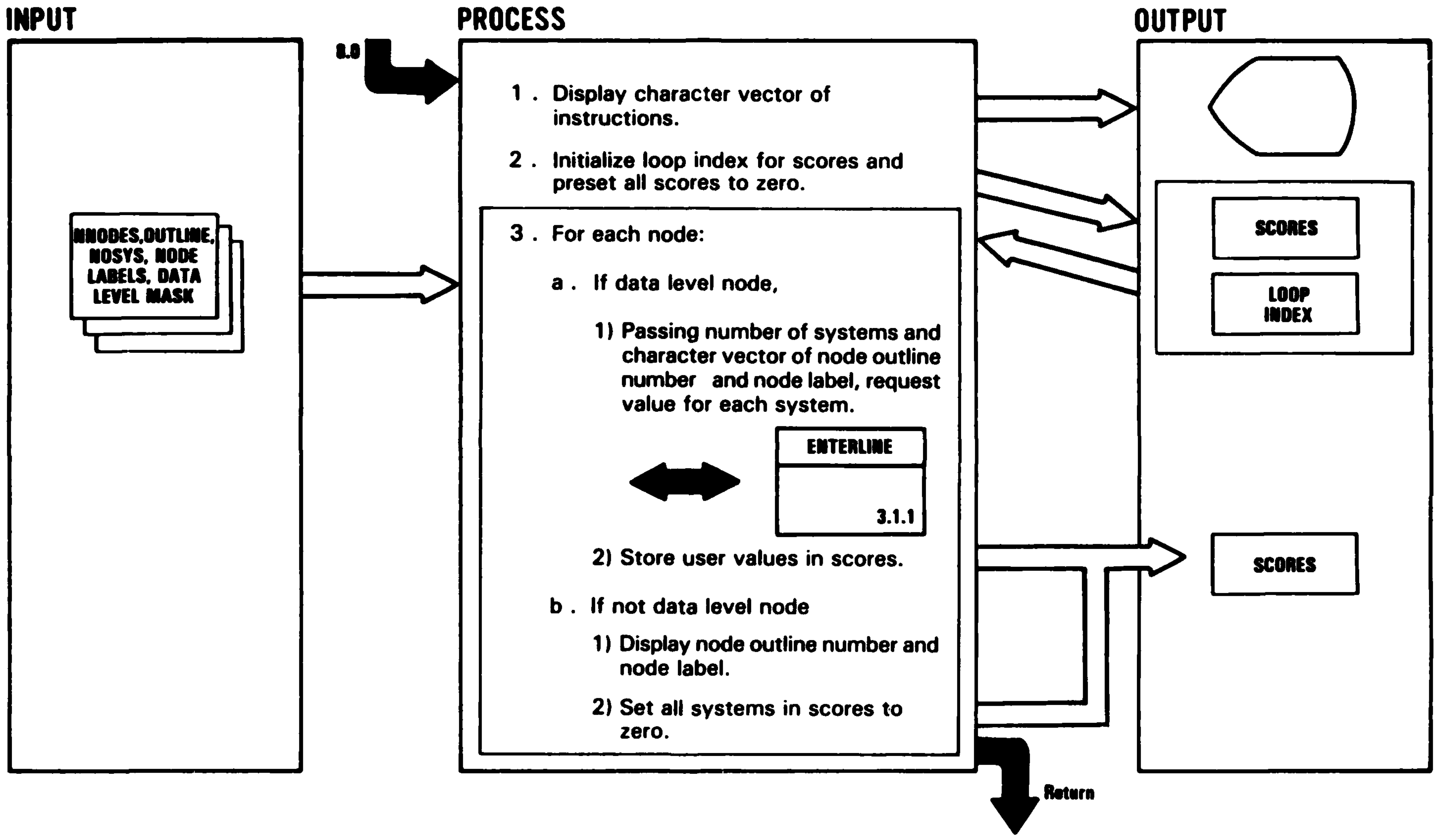


4 . When no option selected display message that model is being recalculated.

5 . Calculate model.

System/Program: RUN Name: LOADSCORE

Diagram ID: 8.1 Description Enter All Scores Page: of



1 . Display character vector of instructions.

2 . Initialize loop index for scores and preset all scores to zero.

3 . For each node:

a . If data level node,

1) Passing number of systems and character vector of node outline number and node label, request value for each system.

2) Store user values in scores.

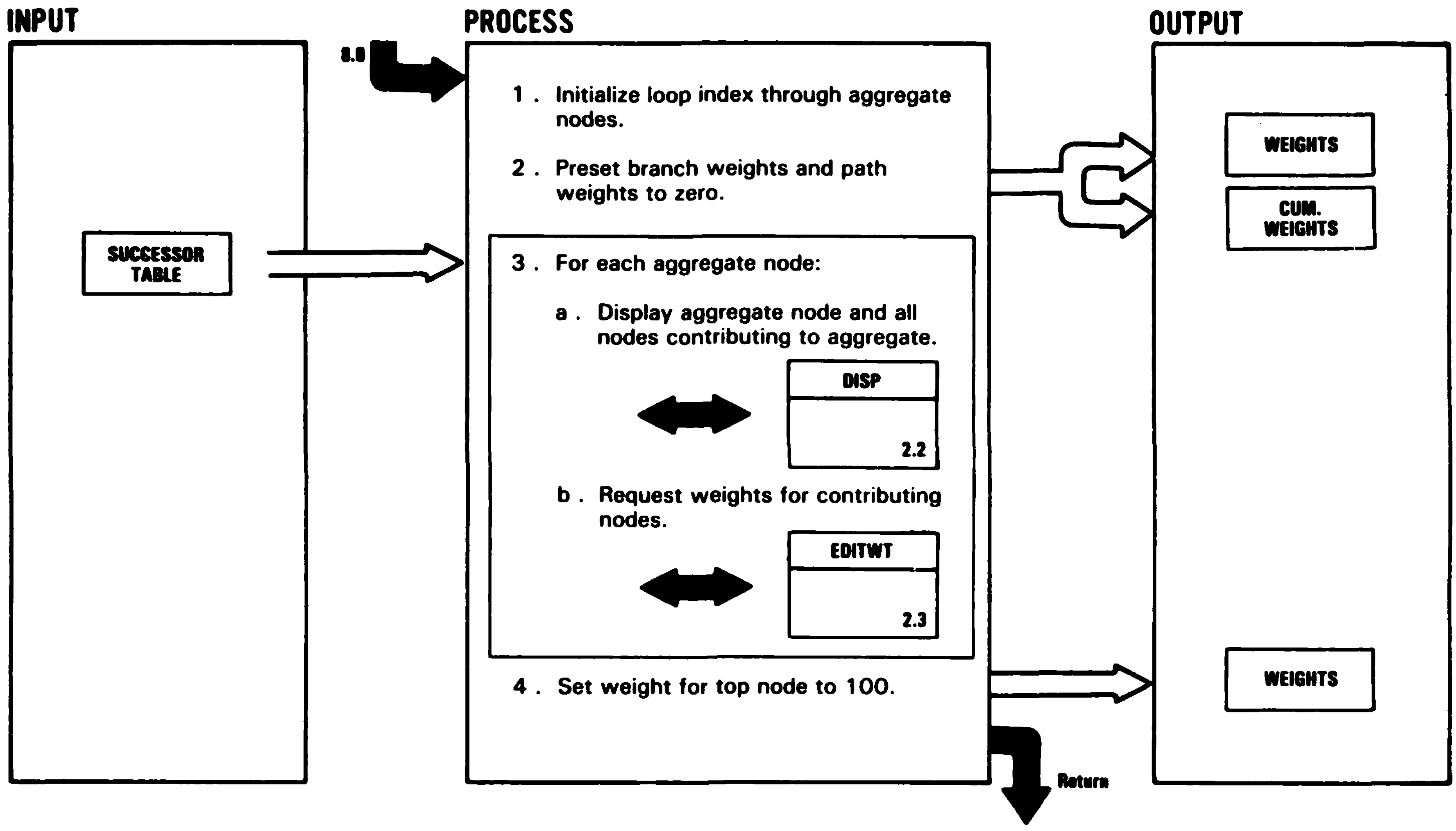
b . If not data level node

1) Display node outline number and node label.

2) Set all systems in scores to zero.

System/Program: RUN Name: NEWWEIGHT

Diagram ID: 8.2 Description Enter All Weights Page: of



1 . Initialize loop index through aggregate nodes.

2 . Preset branch weights and path weights to zero.

3 . For each aggregate node:

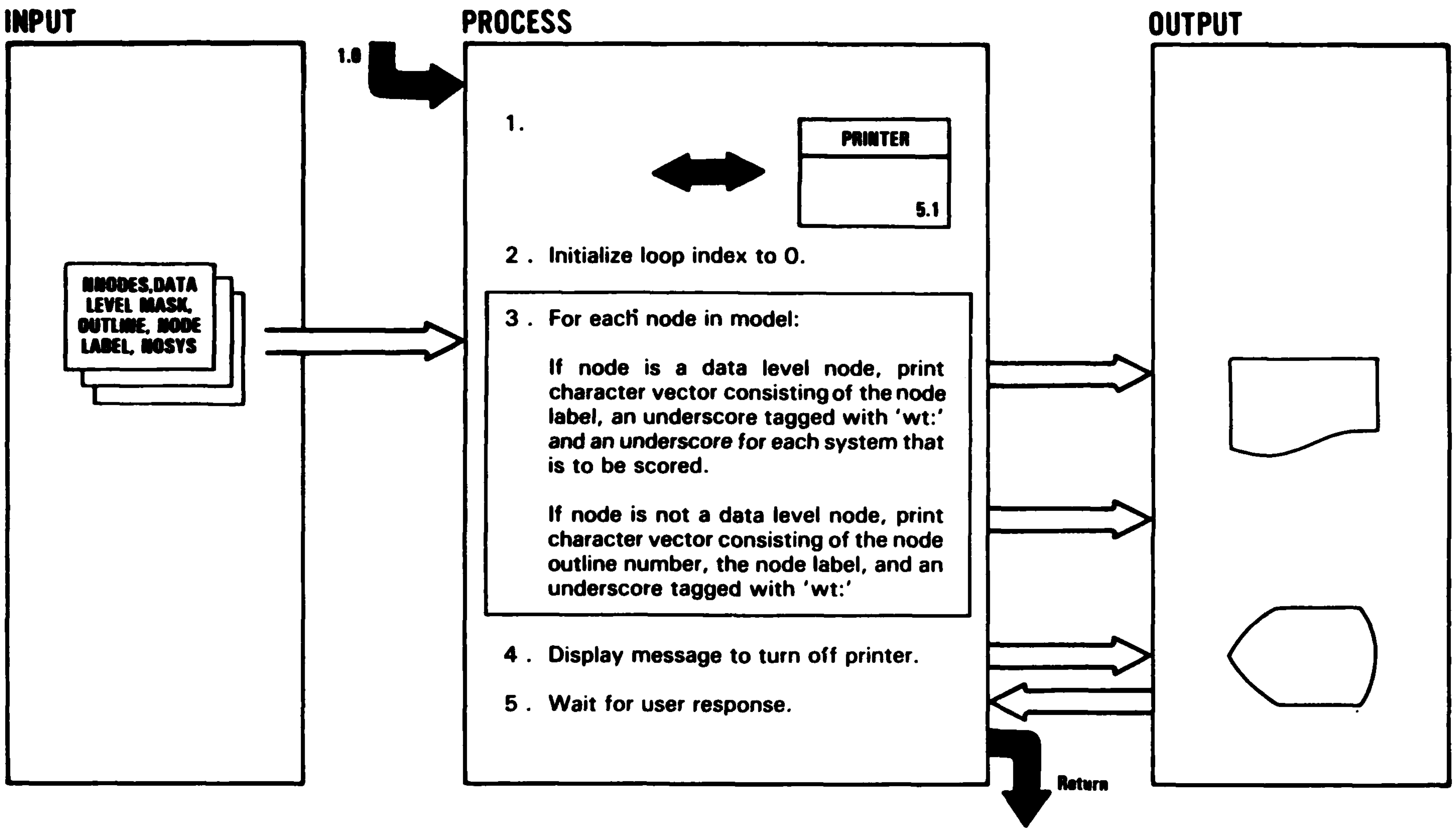
a . Display aggregate node and all nodes contributing to aggregate.

b . Request weights for contributing nodes.

4 . Set weight for top node to 100.

System/Program: RUN  Name: DATASHEET

Diagram ID: 9.0  Description Print Data Sheet  Page: ___ of ___



1.

2. Initialize loop index to 0.

3. For each node in model:

   If node is a data level node, print character vector consisting of the node label, an underscore tagged with 'wt:' and an underscore for each system that is to be scored.

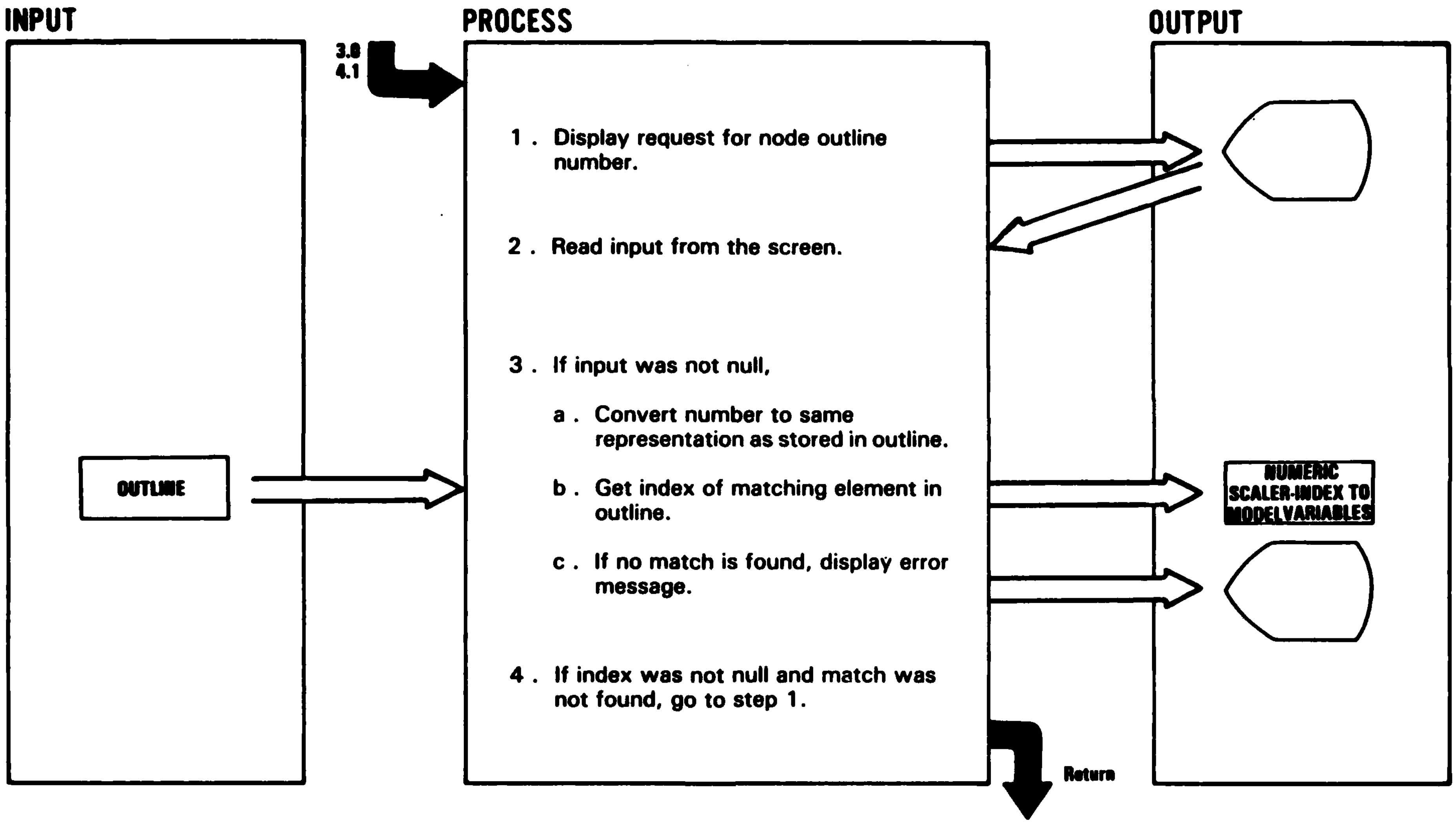
   If node is not a data level node, print character vector consisting of the node outline number, the node label, and an underscore tagged with 'wt:'

4. Display message to turn off printer.

5. Wait for user response.

System/Program: RUN Name: LOCATE

Diagram ID: 10.1 Description Elicit Node Number Page: ____ of ____



1 . Display request for node outline number.

2 . Read input from the screen.

3 . If input was not null,

a . Convert number to same representation as stored in outline.

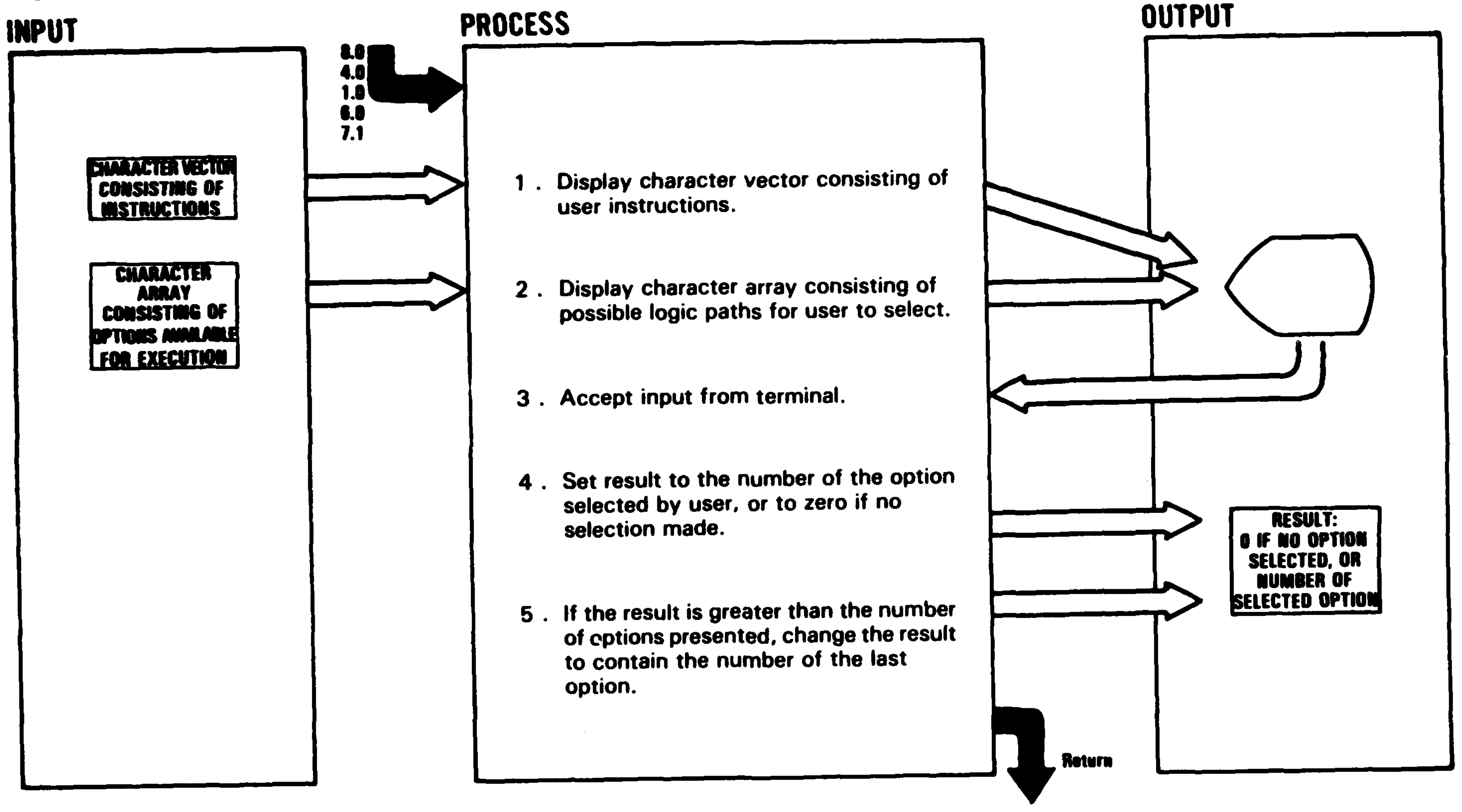
b . Get index of matching element in outline.

c . If no match is found, display error message.

4 . If index was not null and match was not found, go to step 1.

System/Program: RUN Name: MENU

Diagram ID: 10.2 Description Present User Options Page: ___ of ___


INPUT
PROCESS
OUTPUT
8.0
4.0
1.0
6.0
7.1
CHARACTER VECTOR CONSISTING OF INSTRUCTIONS
CHARACTER ARRAY CONSISTING OF OPTIONS AVAILABLE FOR EXECUTION
RESULT: 0 IF NO OPTION SELECTED, OR NUMBER OF SELECTED OPTION
Return

1 . Display character vector consisting of user instructions.

2 . Display character array consisting of possible logic paths for user to select.

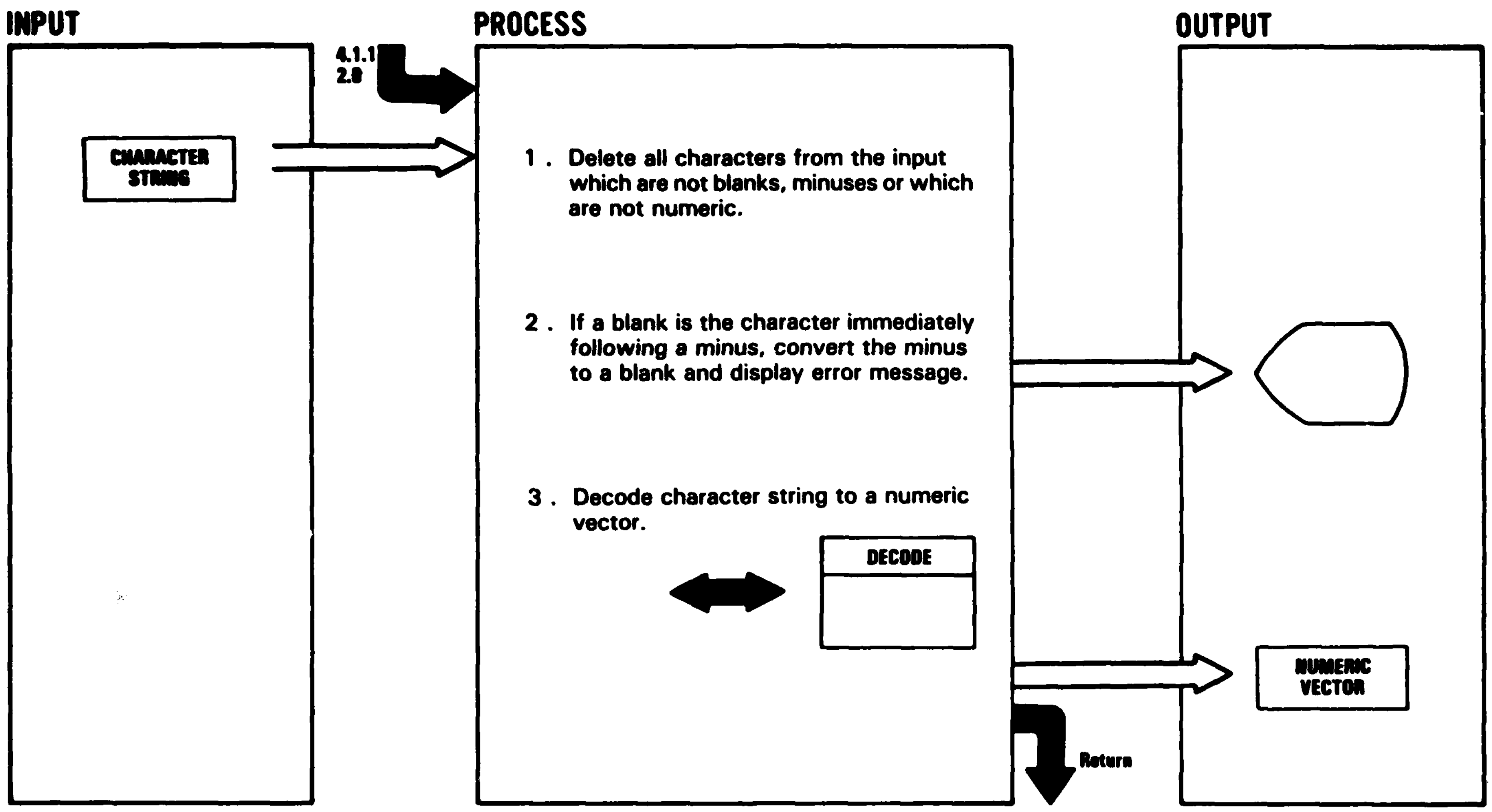
3 . Accept input from terminal.

4 . Set result to the number of the option selected by user, or to zero if no selection made.

5 . If the result is greater than the number of cptions presented, change the result to contain the number of the last option.

System/Program: RUN Name: NUMBERSONLY

Diagram ID: 10.3 Description Convert Character Numbers to Numeric Page: ___ of ___



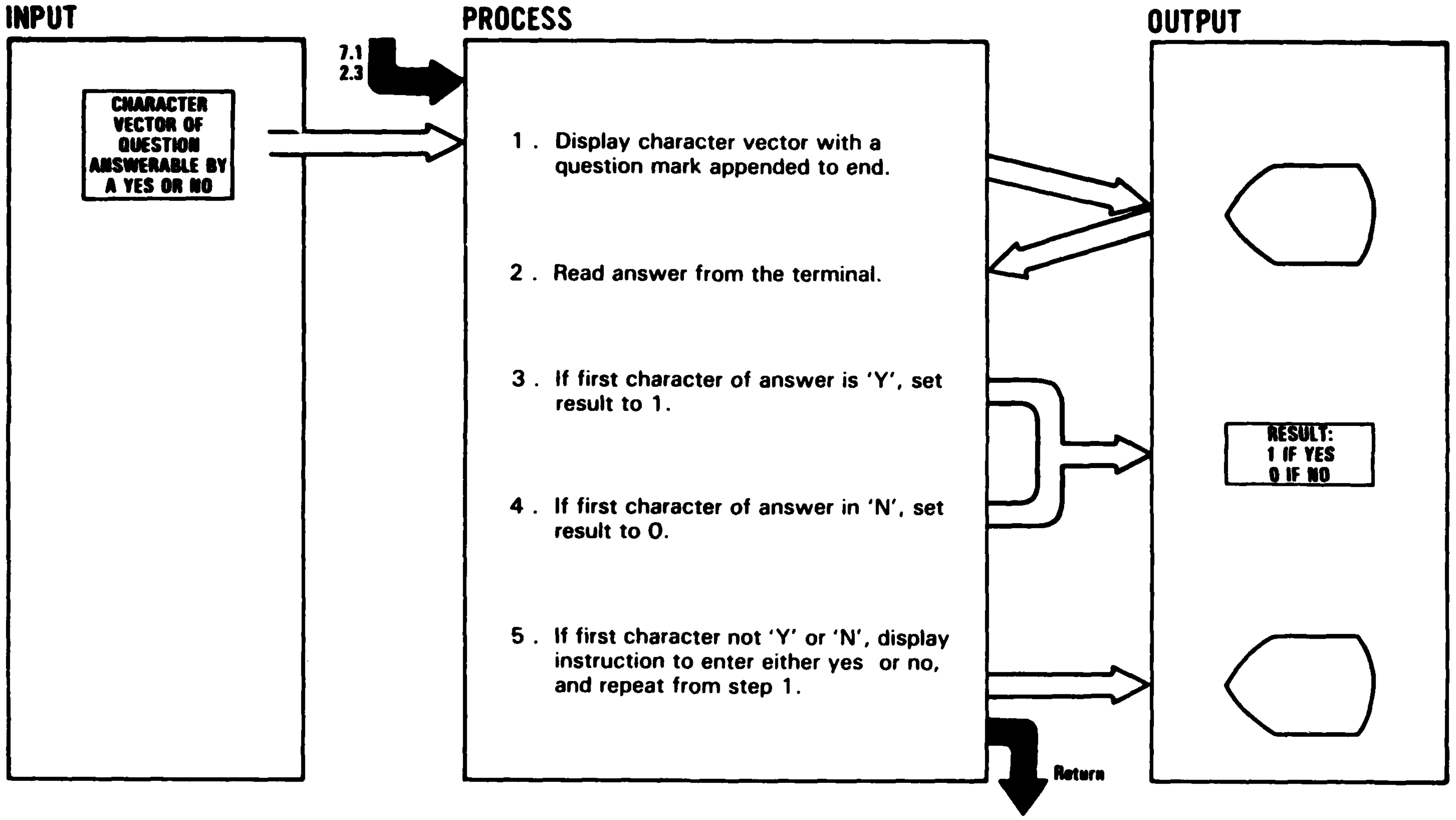
1 . Delete all characters from the input which are not blanks, minuses or which are not numeric.

2 . If a blank is the character immediately following a minus, convert the minus to a blank and display error message.

3 . Decode character string to a numeric vector.

System/Program: RUN Name: YESNO

Diagram ID: 10.4 Description Page: of



1 . Display character vector with a question mark appended to end.

2 . Read answer from the terminal.

3 . If first character of answer is 'Y', set result to 1.

4 . If first character of answer in 'N', set result to 0.

5 . If first character not 'Y' or 'N', display instruction to enter either yes or no, and repeat from step 1.